Semantic Document Derendering: SVG Reconstruction via Vision-Language Modeling

Adam Hazimeh[1], **Ke Wang**[1], **Mark Collier**[2*], **Gilles Baechler**[2*]
Efi Kokiopoulou[2*], **Pascal Frossard**[1]

[1]EPFL
[2]Google DeepMind
adam.hazimeh@epfl.ch

Abstract

Multimedia documents such as slide presentations and posters are designed to be interactive and easy to modify. Yet, they are often distributed in a static raster format, which limits editing and customization. Restoring their editability requires converting these raster images back into structured vector formats. However, existing geometric raster vectorization methods, which rely on low-level primitives like curves and polygons, fall short at this task. Specifically, when applied to complex documents like slides, they fail to preserve the high-level structure, resulting in a flat collection of shapes where the semantic distinction between image and text elements is lost. To overcome this limitation, we address the problem of *semantic document derendering* by introducing `SliDer`, a novel framework that uses Vision-Language Models (VLMs) to derender slide images as compact and editable Scalable Vector Graphic (SVG) representations. `SliDer` detects and extracts the attributes from individual image and text elements in a raster input and organizes them into a coherent SVG format. Crucially, the model iteratively refines its predictions during inference in a process analogous to human design, generating SVG code that more faithfully reconstructs the original raster upon rendering. Furthermore, we introduce `Slide2SVG`, a novel dataset comprising raster-SVG pairs of slide documents curated from real-world scientific presentations, to facilitate future research in this domain. Our results demonstrate that `SliDer` achieves a reconstruction LPIPS of 0.069, and is favored by human evaluators in 82.9% of cases compared to the strongest zero-shot VLM baseline.

Project — https://github.com/adamhazimeh/SliDer

1 Introduction

Digital multimedia documents are often available as raster images, a format that conceals their underlying structure and hinders editability. To edit such documents, we must apply *document derendering*, a process that first recovers the original layout from the pixel-based representation, and then parses identified assets like images and text, to semantically reconstruct the document into an editable form. This enables quick, accessible editing without the need to re-design documents from scratch.

Among common structured representation methods, Scalable Vector Graphics (SVGs) offer a flexible structure for representing multimedia documents by encoding image and text assets as discrete and editable elements. Its hierarchical design allows for precise manipulation of individual components, facilitating straightforward editing and reordering.

Despite the advantages of SVG, most existing approaches that derender raster images into SVG format rely on low-level geometric primitives, such as curves and polygons (Ma et al. 2022; Rodriguez et al. 2023; Carlier et al. 2020; Reddy et al. 2021), which work well for simple icons and logos but fall short when applied to complex multimedia documents. These methods often produce unstructured representations that fail to capture the semantic layout of documents like slides, underscoring the need for SVG reconstruction techniques for multimedia documents that move beyond primitive-based approximations.

Overcoming these limitations requires a model that can interpret intricate visual inputs and generate structured code, a combination of capabilities that constitutes a core strength of modern Vision-Language Models (VLMs). Recent advancements in VLMs have showcased robust performance in code generation (Jiang et al. 2024; Zheng et al. 2023) and image-to-text tasks (Team et al. 2023; Achiam et al. 2023; Bai et al. 2023; Team et al. 2025), demonstrating powerful image understanding and object detection abilities that are well-suited for high-level SVG reconstruction.

Motivated by these successes, we present **`SliDer`** (**Sli**de **Der**enderer), a novel VLM-based framework that converts raster multimedia documents into structured, editable SVG representations. We focus on slide-based documents, as their rich composition of text, images, and complex layouts makes them both a popular communication tool in many domains and a challenging benchmark. As illustrated in Figure 1, our method derenders a raster slide into an SVG repre-

*Google DeepMind contributed in an advisory capacity only. No experiments or research were carried out by Google DeepMind.

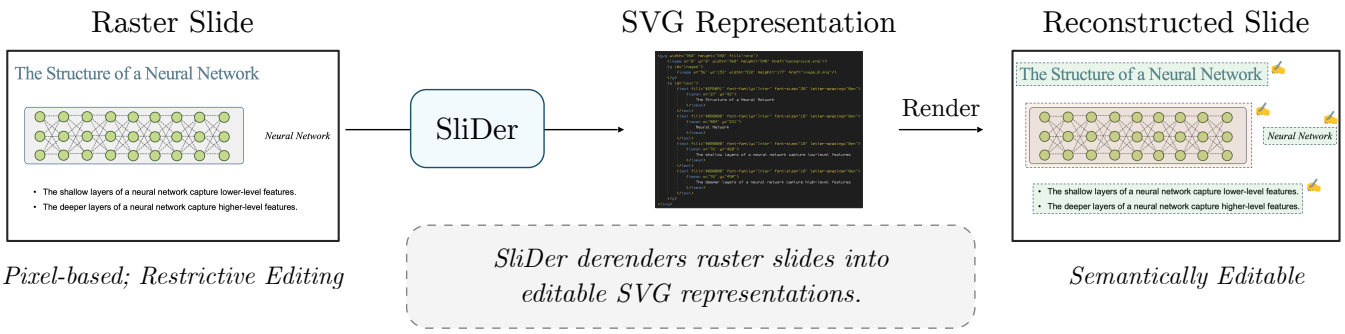

Figure 1: `SliDer` derenders raster slides into editable SVG-based format, allowing flexible editing on the slide such as adjusting figures, modifying text, etc.

sentation that faithfully reconstructs the original raster slide upon rendering. A key feature of our approach is its ability to iteratively refine its own predictions at inference time, allowing it to correct initial errors and progressively improve reconstruction fidelity. Notably, the images and text contained in the raster slide are parsed into individual, editable assets, enabling independent modifications.

To develop our method and advance research in this domain, we also introduce **`Slide2SVG`**, a new dataset for slide derendering. Comprising approximately 38,000 samples collected from real-world scientific presentations, it spans a wide array of designs, content, and layouts, providing a robust foundation for future work in structured document reconstruction.

Using `Slide2SVG`, we evaluate `SliDer` with quantitative metrics and human judgments, focusing on the visual fidelity of its reconstructions. In pairwise tests, human evaluators chose Gemini-based `SliDer` over the strongest zero-shot VLM baseline, GPT-4o (Hurst et al. 2024), in 82.9% of cases and over LIVE (Ma et al. 2022), a leading raster vectorization method, in 91.8%. Perceptual metrics also support this preference: `SliDer` achieves an LPIPS[1] of 0.069 compared to 0.118 and 0.169 for GPT-4o and LIVE, respectively, significantly reducing the perceptual distance between the original raster and the reconstruction.

The primary contributions of our work are as follows[2]:

- We formulate the task of *semantic document derendering*, which involves extracting the overall layout of a multimedia document and parsing each individual asset into an editable format, eventually transforming the raster document into a structured, editable representation.
- We propose **`SliDer`**, a VLM-based framework that iteratively converts raster slides into structured SVG representations, faithfully reconstructing the original slides upon rendering.
- We introduce **`Slide2SVG`**, a novel dataset containing raster slides and their compact SVG representations, to address the shortage of image-to-SVG datasets for multimedia documents.
- We demonstrate through comprehensive quantitative and human evaluations on `Slide2SVG`, that `SliDer` consistently surpasses strong zero-shot VLM and raster vectorization baselines in reconstruction fidelity.

[1]LPIPS is a learned perceptual similarity metric in which lower values indicate higher visual similarity.

[2]Google DeepMind contributed in an advisory capacity only. No experiments or research were carried out by Google DeepMind.

2 Related Work

We briefly survey work on vision-language models, raster vectorization, and document datasets most relevant to our setting, and refer the reader to the Appendix for an extended overview.

2.1 Vision-Language Models

Large Vision-Language Models (VLMs) have shown strong performance in image-to-text generation and visual reasoning (Li et al. 2025; Zhang et al. 2024; Hu et al. 2022; Xie et al. 2022; Hartsock and Rasool 2024; Lee et al. 2024), including visual document understanding (Li et al. 2024; Luo et al. 2022). Because they can both parse complex layouts and generate structured code (Jiang et al. 2024; Zheng et al. 2023), they are a natural fit for SVG-based document derendering.

2.2 Raster Vectorization

Classical methods vectorize rasters via segmentation and diffusion curves (Selinger 2003; Xia, Liao, and Yu 2009; Orzan et al. 2008; Xie et al. 2014), while more recent deep models such as DeepSVG, SVG-VAE, LIVE, Im2Vec, VectorFusion, and StarVector (Carlier et al. 2020; Lopes et al. 2019; Ma et al. 2022; Reddy et al. 2021; Jain, Xie, and Abbeel 2023; Rodriguez et al. 2023) learn to generate or refine vector primitives. However, they typically output flat sets of paths and curves rather than a structured hierarchy of editable document elements, limiting their suitability for our task.

2.3 Datasets for SVG Generation and Document Understanding

Existing SVG generation datasets mostly target simple graphics such as icons or emojis (Cai et al. 2023; Wu et al. 2023; Reddy et al. 2021; Rodriguez et al. 2023; Cao et al.

2023), and thus lack the layout complexity of real documents. Conversely, document understanding benchmarks like DocLayNet, PubLayNet, SlideVQA, and DocSynth (Pfitzmann et al. 2022; Zhong, Tang, and Yepes 2019; Tanaka et al. 2023; Zhao et al. 2024) focus on layout analysis but do not provide full, editable SVG representations. Our `Slide2SVG` dataset is designed to bridge this gap.

3 Background and Problem Formulation

3.1 Representing Slides in SVG Format

SVG offers a structured way to represent slide images by describing content as layered, discrete assets rather than as a dense array of pixels. It allows image and text elements to be encoded as individual objects with clearly defined attributes, providing layout-informed editability. For instance, image assets can be stored as external files referenced within the SVG, with attributes specifying their coordinates, width, and height. Similarly, text assets can be embedded directly into the SVG and include attributes such as the text content, position, font size, font family, color, and more.

This asset-based format provides fine-grained control over slide content: users can easily reposition figures, update text, and adjust layouts to meet evolving design requirements, making SVG an ideal candidate for semantic document derendering.

3.2 Slide Derendering Problem Formulation

Slide derendering transforms a raster slide into a structured, editable SVG representation. This task poses three primary challenges:

- **Asset Identification:** Detecting and identifying individual elements, such as images and text, even in complex/overlapping layouts.
- **Attribute Inference:** Correctly determining each asset's attributes, including spatial coordinates and stylistic features.
- **SVG Code Generation:** Converting the inferred structure and attributes of identified assets into valid SVG code that faithfully replicates the original slide when rendered and supports further editing.

The ultimate goal of slide derendering is to produce SVG code that achieves two complementary objectives: faithfully replicating the visual design of the original slide, while representing the slide in a compact, easy-to-edit vector format.

4 `SliDer`: A VLM-Based Framework for Slide Derendering

To this end, we propose **`SliDer`**, a VLM-based approach that tackles the problem of slide derendering, converting raster slides into structured and editable SVG representations. It utilizes the visual understanding ability of VLMs to interpret the complex layout of the raster slide, identify the individual image and text assets, and extract their respective attributes. The extracted information is then organized by the VLM to generate a final SVG representation of the input raster slide.

Importantly, derendering a raster slide is a complex task that requires accurately predicting the spatial attributes of individual assets, which is often difficult to accomplish in a single round of inference. To address this challenge, we integrate an *iterative refinement* step into our framework. This enables the model to progressively improve its own predictions at inference time, correcting initial errors to achieve better reconstruction quality.

4.1 Input Representation

In our framework, the VLM takes three primary inputs during both training and inference: a raster slide, an instruction prompt, and an auxiliary SVG context.

Raster Slide The primary visual input is the raster slide that is to be derendered. Typically, a slide is composed of three types of assets:

- *Text boxes*, which represent text content as well as spatial (x, y, width, height) and stylistic attributes, including font size, font family, color, and letter spacing.
- *Images*, e.g., figures, which contain RGB image content and spatial attributes. In the target SVG, image content is represented using an `<href>` tag pointing to an external image file.
- *Background image*, which is assumed to stretch the canvas fully and is treated similarly to other image assets.

Derendering Instructions A text prompt provides high-level guidance, instructing the VLM to generate SVG code. The specific instruction used is:

> "De-render this raster image: `<image>`. You may find the provided SVG template useful: `<Auxiliary SVG Context>`"

Auxiliary SVG Context To guide the VLM's prediction and enable iterative refinement, an SVG context is provided as auxiliary information. This context takes one of three forms, each serving a distinct purpose in training:

- **Skeleton Template**: This is a bare-bones SVG structure containing no specific content, spatial, or stylistic attributes. Its purpose is to train the model to generate a complete SVG representation from scratch.
- **Partial Template**: This template contains only the spatial attributes (i.e., bounding box coordinates) for all text and image assets, with all stylistic attributes left empty. By providing the layout, this context focuses the model's task on learning to infer stylistic properties.
- **Initial Prediction**: This is a valid SVG representation generated by the same model architecture trained in a separate run, representing a first-pass prediction. It contains potentially imperfect spatial and stylistic attributes, and is crucial for training the model to perform iterative refinement by learning to correct its prior mistakes.

4.2 Training Process

Based on the aforementioned inputs, a pre-trained, general-use VLM is fine-tuned to generate a complete SVG representation that faithfully reconstructs the provided raster

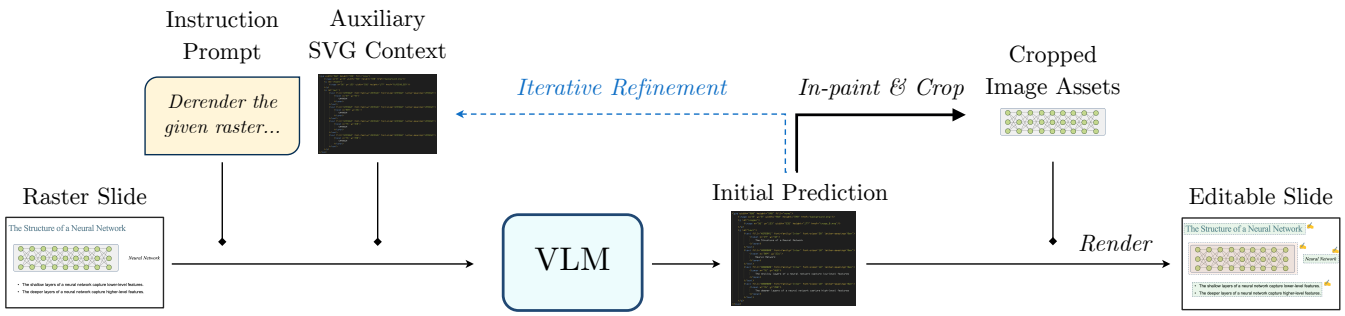

Figure 2: Overview of the SliDer inference pipeline. The VLM takes as input a raster slide, an instruction prompt, and an auxiliary SVG context, generating an editable SVG representation. The generated SVG can optionally be fed back to the VLM for iterative refinement. Given the final predicted SVG, the bounding box information is extracted to crop the image assets from the original raster into external PNG files. Finally, the slide is reconstructed by rendering the resulting SVG code.

image. The model produces the entire SVG code in a single generation step, by simultaneously predicting the spatial placement of image assets, which are hyper-referenced in the SVG code using <image> tags, and the content and styling of text assets, leveraging its embedded OCR ability.

To ensure the model is robust and can handle different scenarios at inference time, we employ a data augmentation strategy that leverages the diverse input contexts described in Section 4.1. For raster slides in the training set, we create three distinct training variants by pairing them with different SVG contexts:

- A *skeleton template*, to learn generation from scratch.
- A *partial template*, where bounding boxes are generated by an external YOLO-based object detector (Jocher, Chaurasia, and Qiu 2023).
- An *initial prediction* (valid SVG), generated by a separately trained VLM. The starting context to obtain the initial prediction can either be a skeleton template, or a YOLO-guided partial template.

More details about the external models used are provided in the Appendix.

4.3 Inference Process

At inference time, SliDer follows a multi-step process to transform a raster slide into a fully editable SVG. The process begins with an initial generation pass, which can optionally be enhanced through iterative refinement, and concludes with a final post-processing step to extract all assets.

Initial SVG Generation The process starts by feeding the VLM the input raster slide and an initial SVG context. This context is typically either a *skeleton template* for generating the SVG from scratch, or a *partial template* (i.e., with bounding boxes from an external object detector) to leverage prior spatial information.

Iterative Refinement Complex slides with intricate layouts or subtle stylistic details can pose a challenge for any single-pass generation model. To enhance derendering quality in these cases, SliDer includes an optional iterative refinement capability. As shown in Figure 2, the SVG generated in the previous step can be fed back into the model as an *initial* context, along with the original raster slide. This process allows the model to polish its initial prediction, addressing misalignments, stylistic inconsistencies, or layout errors.

Post-processing Once the final SVG representation is generated, we perform two post-processing steps to produce the final derendering.

- **Background and Overlap Resolution:** This step resolves occlusions and isolates the background. If the predicted SVG bounding boxes for any assets overlap, we employ the TELEA (Telea 2004) inpainting algorithm to fill in the occluded regions. To extract the background, we mask out all foreground assets from the original raster and use the same inpainting technique to fill the resulting empty areas, producing a clean background image.
- **Image Asset Extraction:** After the inpainting step, all image assets defined in the final SVG are cropped from the original raster slide using their predicted bounding boxes. These cropped assets are then saved as external PNG files with filenames that correspond to those hyper-referenced in the generated SVG code (e.g., 'image_1.png').

The final output of this entire process is a fully editable SVG file, accompanied by all associated image assets as standalone files.

5 Slide2SVG: A New Dataset for Slide Derendering

As established in Section 2, existing datasets for vector graphics and document understanding are ill-suited for the task of semantic document derendering. To fill this critical gap, we introduce **Slide2SVG**, a new real-world dataset designed to facilitate the conversion of rasterized slides into structured, editable SVG representations. Curated from publicly available conference slides, the dataset captures diverse design styles, font choices, image placements, and layout configurations found in real-world slides, offering a challenging yet realistic platform for evaluating derendering pipelines. Each sample in Slide2SVG is composed of:

- The original raster slide in PNG format.

		mIoU (%) ↑	OCR Accuracy (%) ↑	Visual metrics			Elo ↑
				MSE ↓	LPIPS ↓	CLIP Sim. ↑	
Raster Vectorization	LIVE	N/A	N/A	18.19	0.169	0.7823	794
Zero-shot VLMs	GPT-4o	88.42	69.82	14.48	0.118	0.883	948
	Gemma	80.28	65.57	15.67	0.150	0.848	880
	Gemini	83.78	67.71	15.05	0.123	0.879	925
`SliDer`	Gemma	**89.36**	**93.53**	13.38	0.075	0.950	1207
	Gemini	89.14	92.85	**13.30**	**0.069**	**0.953**	**1245**

Table 1: Quantitative evaluation of different derendering methods. Zero-shot methods use YOLO-guided partial templates, with no iterative refinement, while results for `SliDer` are reported with iterative refinement. mIoU and OCR Accuracy are not available for LIVE since it only generates vector paths. Bold/underlined values correspond to best/second-best per metric.

- The corresponding SVG representation of the raster slide, where text and image assets are encoded compactly to preserve editability.
- Individual image assets referenced in the SVG code, separately accessible as PNG files.

To construct this dataset, we assembled slides from academic conference presentations, particularly within the machine learning community, following a systematic data collection and processing pipeline:

1. **PDF Collection** – We collect presentation slides in PDF format from the archives of several major machine learning conferences.
2. **SVG Conversion** – The PDFs are then converted to Figma designs (Figma, Inc. 2025) and exported as raw SVG files. Figma is a web-based design tool primarily used for designing user interfaces and prototypes, but it also integrates community plugins, some of which can be used to convert PDFs to Figma designs. Note that this conversion is only used to build `Slide2SVG`. At inference time, we naturally assume that the slide is not available in a vector format (e.g., SVG, PDF) and must be derendered from a raster format (e.g., PNG).
3. **Asset Grouping** – Text assets in the Figma-exported SVG slides are often arbitrarily grouped based on heuristics rather than semantic coherence. To ensure a structured and consistent grouping, we use the zero-shot DocLayout-YOLO model (Zhao et al. 2024) to reorganize text elements. Text assets identified as belonging to a single entity are merged into a unified text element.
4. **Outlier Filtering** – We filter out slides containing more than 8 image assets or 31 text assets (i.e., 95th percentile of asset counts), as they are excessively complex and not representative of common real-world slides.
5. **Rasterization** – The obtained SVG representations are finally rendered into PNG format to obtain the corresponding raster slides.

The final dataset is randomly divided into roughly 38,000 training samples and 225 test samples, each consisting of a raster image and its corresponding SVG representation.

By providing a new standardized dataset for raster-to-SVG conversion, `Slide2SVG` lays a foundation for future research in fields including document understanding and layout generation.

6 Experiments

6.1 Experimental Setup

Models and Training Details We experiment with two large VLMs: Gemini-1.5-Flash (Team et al. 2024) and the open-source Gemma 3 (12B) (Team et al. 2025). We train both models using the proposed `SliDer` framework on the training set of `Slide2SVG`. This framework is designed to train the model to handle diverse scenarios, such as generating SVGs from scratch (using skeleton templates), leveraging spatial priors (using partial templates), or refining previous outputs (using initial predictions). We demonstrate the results of `SliDer` configured with a partial template (bounding box priors) and iterative refinement in Table 1, while presenting other configurations in Table 2.

The bounding box priors used in partial templates are obtained from a YOLOv8 model (Jocher, Chaurasia, and Qiu 2023) trained for 50 epochs on the `Slide2SVG` training set, with the objective of detecting image and text objects in the raster slide. Initial predictions used for iterative refinement are generated by a VLM of the same family. For instance, the Gemini-based `SliDer` is refined using an initial prediction from a separately trained Gemini model. Further experimental details are presented in the Appendix.

Baselines We compare `SliDer` against two categories of baselines:

- *Zero-shot VLMs:* We evaluate the zero-shot performance of Gemini-1.5 Flash, Gemma 3 (12B), and GPT-4o (Hurst et al. 2024). To ensure a fair comparison, these models are also provided with the YOLO-guided partial template, but without iterative refinement [3].
- *Raster Vectorization:* We compare against LIVE (Ma et al. 2022), a deep-learning-based raster vectorization method that generates low-level geometric primitives.

[3] Our empirical results show that zero-shot VLMs do not benefit from iterative refinement.

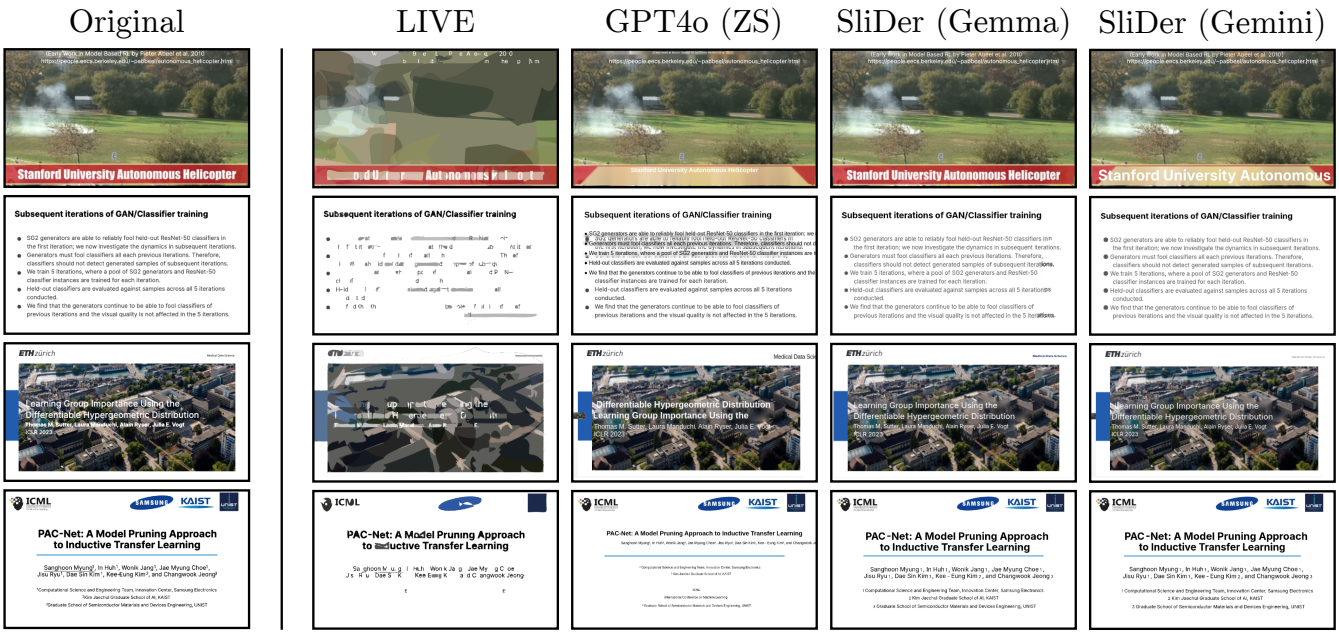

Figure 3: Examples of derendered slide images. Each row contains a separate sample, showing the original raster slide and the reconstructions from the derendered SVGs by different methods. For `SliDer`, we show the YOLO-guided versions with one step of iterative refinement. "ZS" refers to zero-shot methods.

6.2 Evaluation Metrics

To systematically assess model performance, we evaluate the generated SVGs using metrics that measure varying aspects of derendering quality.

- *Bounding Box mIoU*: Measures the spatial alignment of localized assets by the mean Intersection over Union (mIoU) between predicted and ground-truth bounding boxes, averaged over both image and text elements.
- *Text OCR Accuracy*: Evaluates character-level similarity between predicted and ground-truth text, computed by concatenating all text into one string and measuring sequence-level accuracy.
- *Mean Squared Error (MSE)*: Measures pixel-wise reconstruction error (smaller is better). MSE is computed on pixels in the range of [0, 255].
- *CLIP Similarity* (Mayilvahanan et al. 2024): Computes the cosine similarity between image feature embeddings from a CLIP model ($\in$[-1, 1]; higher is better).
- *Learned Perceptual Image Patch Similarity (LPIPS)* (Zhang et al. 2018): Uses deep features from VGG-16 (Simonyan and Zisserman 2015) to quantify human-aligned perceptual distance ($\in$[0, 1]; lower is better).
- *Elo Score*: A rating system reflecting human preference. In our case, methods are compared pairwise, with human evaluators determining the superior output based on visual fidelity. These outcomes are then used to update each method's Elo rating, reflecting their relative performance (higher is better). To compute the Elo score, we collect rankings from 6 human evaluators and use an initial score of 1000 with a K-factor of 4, similar to the setup used in Chatbot Arena (Chiang et al. 2024).

6.3 Main Results

Table 1 presents the main quantitative evaluations. The results shows that our `SliDer` framework improves upon all baselines, with particularly notable gains in content preservation and perceptual quality as judged by both automated metrics and human evaluators.

The human evaluation results, measured by Elo scores, show a clear preference for our method. `SliDer` models score above 1200, significantly higher than 948 for the strongest zero-shot VLM, GPT-4o. This gap reflects that human evaluators favored our Gemini-based `SliDer` variant over GPT-4o in 82.9% of cases, and over the geometric vectorization method (LIVE) in 91.8% of cases. Details about computing the win-rate are presented in the Appendix.

This human preference aligns with the automated perceptual metrics. For instance, `SliDer`-Gemini achieves an LPIPS score of 0.069, a significant reduction compared to 0.118 for the best baseline. This is further supported by a higher CLIP Similarity score (0.953 vs. 0.883).

These perceptual improvements are driven by high fidelity in both content and layout. `SliDer` excels in text extraction, achieving an OCR accuracy above 93%, a substantial increase from 69.82% for the best baseline. This improvement arises because our fine-tuning process teaches the model to perform OCR effectively within the context of predefined bounding boxes. While the gain in layout accuracy (mIoU) is more modest, `SliDer` still outperforms the baselines. As expected, the geometric vectorization method,

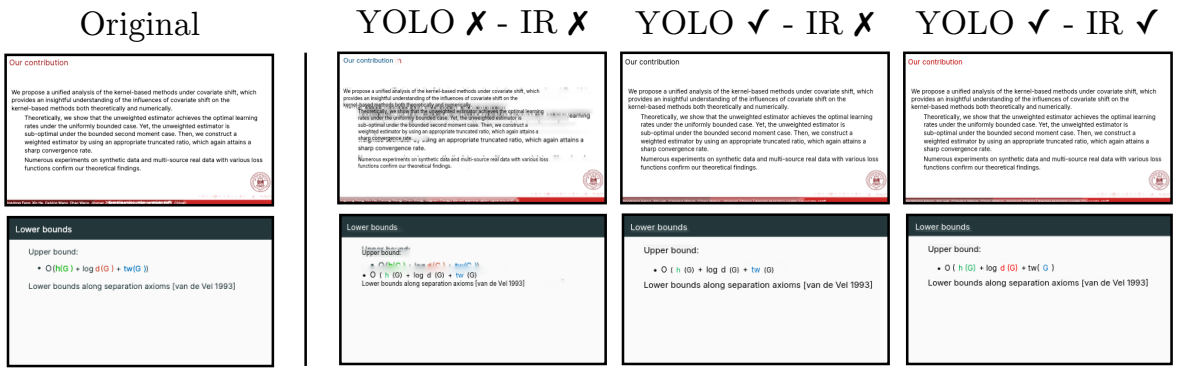

Figure 4: Qualitative examples for the ablations on the effect of bounding box information priors and iterative refinement during inference. We use the Gemini variant of `SliDer`. "YOLO" indicates that the model uses bounding box priors. "IR" indicates that the model performs one step of iterative refinement at inference time.

	YOLO	IR	mIoU↑	OCR↑	Visual metrics		
					MSE↓	LPIPS↓	CLIP↑
SliDer (Gemini)	✗	✗	81.90	89.78	14.47	0.090	0.929
	✓	✗	**89.71**	92.42	13.46	0.072	0.951
	✗	✓	81.70	90.57	14.28	0.088	0.930
	✓	✓	89.14	**92.85**	**13.30**	**0.069**	**0.953**

Table 2: Ablations for Gemini-based `SliDer`. We analyze the effect of adding YOLO-based bounding box priors (YOLO) and one step of iterative refinement (IR). mIoU and OCR accuracy are reported as percentages (%).

LIVE, is not competitive on this complex, multi-element derendering task.

6.4 Ablation Studies

We conduct ablation studies on the Gemini-based `SliDer` model to isolate the effects of our framework's key components. The results are shown in Table 2.

Effect of Prior Bounding Box Information Providing YOLO-based bounding box priors is shown to be effective for achieving a high-quality layout foundation. Comparing the model with priors (row 2) versus without (row 1), the mIoU jumps from 81.90% to a stronger 89.71%. This improved spatial localization has a cascading positive effect on other metrics, improving OCR accuracy from 89.78% to 92.42% and LPIPS from 0.090 to 0.072. This validates that guiding the model with a reliable layout allows it to produce superior results, a practical strategy given that such priors can be readily obtained by pre-trained layout detection models like DocLayout-YOLO (Zhao et al. 2024).

Effect of Iterative Refinement One step of iterative refinement provides a consistent boost in performance, acting as a final polishing step. When applied to the model with YOLO priors (row 4 vs. row 2), we observe improvements across all visual metrics, with LPIPS dropping from 0.072 to 0.069. Even without priors (row 3 vs. row 1), refinement improves OCR accuracy and visual metrics. The best overall performance is achieved by combining both priors and refinement, validating our main model configuration choice.

6.5 Qualitative Analysis

Figure 3 provides a qualitative comparison of `SliDer` against LIVE and GPT-4o, where `SliDer`'s reconstructions closely replicate the original slides. In contrast, LIVE fails to render readable text, while the zero-shot GPT-4o output suffers from text misalignment and stylistic errors. Our method successfully captures fine-grained details, including logos and horizontal rules. These visual gains mirror the quantitative improvements reported in Table 1.

Moreover, Figure 4 qualitatively demonstrates the impact of `SliDer`'s components. Without bounding box priors, text and image elements are visibly misaligned with their original locations. The introduction of YOLO priors corrects these spatial errors, creating a coherent layout. Finally, iterative refinement further reduces remaining stylistic inconsistencies and improves cropping quality. This visual progression confirms that each component plays a crucial role in achieving the final, high-fidelity reconstruction.

7 Conclusion & Future Work

In this work, we presented `SliDer`, a VLM-based framework that transforms rasterized slides into structured, editable SVG representations. Our approach segments content elements and leverages an iterative refinement process to improve reconstruction fidelity. To facilitate this research, we also introduced `Slide2SVG`, a new dataset curated from real-world scientific presentations. Quantitative and human-preference evaluations confirm that `SliDer` produces editable outputs that are visually faithful and preferred over strong zero-shot baselines. Opportunities for future work include improving performance on layouts with higher content density and expanding `SliDer` to support more complex multimedia documents, such as posters and infographics. Further investigation into the trade-offs between derendering quality and the computational cost of multi-step iterative refinement also presents a valuable research direction.

Acknowledgements

We thank Alba Carballo Castro, Alessandro Favero, Amel Abdelraheem, Guillermo Ortiz-Jiménez, Imane Araf, Nikolaos Dimitriadis, Seth Nabarro, and Sevda Ögüt for helpful feedback and discussions.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Cai, M.; Huang, Z.; Li, Y.; Ojha, U.; Wang, H.; and Lee, Y. J. 2023. Leveraging large language models for scalable vector graphics-driven image understanding. *arXiv preprint arXiv:2306.06094*.

Cao, D.; Wang, Z.; Echevarria, J.; and Liu, Y. 2023. Svgformer: Representation learning for continuous vector graphics using transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Carlier, A.; Danelljan, M.; Alahi, A.; and Timofte, R. 2020. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems (NeurIPS)*.

Chiang, W.-L.; Zheng, L.; Sheng, Y.; Angelopoulos, A. N.; Li, T.; Li, D.; Zhu, B.; Zhang, H.; Jordan, M. I.; Gonzalez, J. E.; and Stoica, I. 2024. Chatbot arena: an open platform for evaluating LLMs by human preference. In *International Conference on Machine Learning*.

Figma, Inc. 2025. Figma - The Collaborative Interface Design Tool. Accessed: 2025-03-06.

Hartsock, I.; and Rasool, G. 2024. Vision-language models for medical report generation and visual question answering: A review. *Frontiers in Artificial Intelligence*, 7.

Hu, X.; Gan, Z.; Wang, J.; Yang, Z.; Liu, Z.; Lu, Y.; and Wang, L. 2022. Scaling up vision-language pre-training for image captioning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; Hayes, A.; Radford, A.; et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jain, A.; Xie, A.; and Abbeel, P. 2023. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jiang, J.; Wang, F.; Shen, J.; Kim, S.; and Kim, S. 2024. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

Jocher, G.; Chaurasia, A.; and Qiu, J. 2023. Ultralytics YOLOv8.

Lee, J.; Cha, S.; Lee, Y.; and Yang, C. 2024. Visual question answering instruction: Unlocking multimodal large language model to domain-specific visual multitasks. *arXiv preprint arXiv:2402.08360*.

Li, X.; Wu, Y.; Jiang, X.; Guo, Z.; Gong, M.; Cao, H.; Liu, Y.; Jiang, D.; and Sun, X. 2024. Enhancing visual document understanding with contrastive learning in large visual-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, Z.; Wu, X.; Du, H.; Nghiem, H.; and Shi, G. 2025. Benchmark evaluations, applications, and challenges of large vision language models: A survey. *arXiv preprint arXiv:2501.02189*.

Lopes, R. G.; Ha, D.; Eck, D.; and Shlens, J. 2019. A learned representation for scalable vector graphics. In *IEEE/CVF International Conference on Computer Vision*.

Luo, C.; Tang, G.; Zheng, Q.; Yao, C.; Jin, L.; Li, C.; Xue, Y.; and Si, L. 2022. Bi-vldoc: Bidirectional vision-language modeling for visually-rich document understanding. *arXiv preprint arXiv:2206.13155*.

Ma, X.; Zhou, Y.; Xu, X.; Sun, B.; Filev, V.; Orlov, N.; Fu, Y.; and Shi, H. 2022. Towards layer-wise image vectorization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mayilvahanan, P.; Wiedemer, T.; Rusak, E.; Bethge, M.; and Brendel, W. 2024. Does CLIP's generalization performance mainly stem from high train-test similarity? In *International Conference on Learning Representations (ICLR)*.

Orzan, A.; Bousseau, A.; Winnemöller, H.; Barla, P.; Thollot, J.; and Salesin, D. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics (TOG)*.

Pfitzmann, B.; Auer, C.; Dolfi, M.; Nassar, A. S.; and Staar, P. 2022. Doclaynet: A large human-annotated dataset for document-layout segmentation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Reddy, P.; Gharbi, M.; Lukac, M.; and Mitra, N. J. 2021. Im2Vec: Synthesizing Vector Graphics without Vector Supervision. *arXiv preprint arXiv:2102.02798*.

Rodriguez, J. A.; Agarwal, S.; Laradji, I. H.; Rodriguez, P.; Vazquez, D.; Pal, C.; and Pedersoli, M. 2023. Starvector: Generating scalable vector graphics code from images. *arXiv preprint arXiv:2312.11556*.

Selinger, P. 2003. Potrace: a polygon-based tracing algorithm.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Bengio, Y.; and LeCun, Y., eds., *International Conference on Learning Representations (ICLR)*.

Tanaka, R.; Nishida, K.; Nishida, K.; Hasegawa, T.; Saito, I.; and Saito, K. 2023. Slidevqa: A dataset for document visual question answering on multiple images. In *AAAI Conference on Artificial Intelligence*.

Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; Millican, K.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Team, G.; Georgiev, P.; Lei, V. I.; Burnell, R.; Bai, L.; Gulati, A.; Tanzer, G.; Vincent, D.; Pan, Z.; Wang, S.;

et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Team, G.; Kamath, A.; Ferret, J.; Pathak, S.; Vieillard, N.; Merhej, R.; Perrin, S.; Matejovicova, T.; Ramé, A.; Rivière, M.; et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Telea, A. 2004. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1): 23–34.

Wu, R.; Su, W.; Ma, K.; and Liao, J. 2023. Iconshop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics (TOG)*.

Xia, T.; Liao, B.; and Yu, Y. 2009. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics (TOG)*.

Xie, G.; Sun, X.; Tong, X.; and Nowrouzezahrai, D. 2014. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Transactions on Graphics (TOG)*.

Xie, Y.; Zhou, L.; Dai, X.; Yuan, L.; Bach, N.; Liu, C.; and Zeng, M. 2022. Visual clues: Bridging vision and language foundations for image paragraph captioning. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, J.; Huang, J.; Jin, S.; and Lu, S. 2024. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhao, Z.; Kang, H.; Wang, B.; and He, C. 2024. Doclayout-yolo: Enhancing document layout analysis through diverse synthetic data and global-to-local adaptive perception. *arXiv preprint arXiv:2410.12628*.

Zheng, Z.; Ning, K.; Wang, Y.; Zhang, J.; Zheng, D.; Ye, M.; and Chen, J. 2023. A survey of large language models for code: Evolution, benchmarking, and future trends. *arXiv preprint arXiv:2311.10372*.

Zhong, X.; Tang, J.; and Yepes, A. J. 2019. Publaynet: largest dataset ever for document layout analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*.