Group 10: Classifier and Diffusion Model for Presentation Builder
MSML 640 – Final Project Report

Anirud Mohan 121109941 | Javad Baghirov 121093264 | Hsing-Hao Wang 116855587 | William C. Loe 121290311

December 8, 2025

Project Repository: `https://github.com/willloe/MSML640_Group10`

1 Introduction

Slide decks are a primary way to teach, pitch ideas, and summarize projects. However, making slides that are readable and visually appealing is often time consuming and requires design skills which a lot of people do not have. Existing tools offer fixed templates, but these rarely match the actual content structure and still require a lot of manual editing.

Our project investigates whether we can use computer vision and generative models to automate part of this process. We build two major components of an end-to-end pipeline. First one is a classifier pipeline that analyzes slide images, detects and labels key regions (such as title, body text, images, logos and graphs). This pipeline outputs a structured layout representation. Secondly, a diffusion pipeline that uses this layout to generate backgrounds that preserve clean regions for text.

1.1 Problem Statement

We address the problem of automatically improving slide design given only the slide image (or its layout) and the user's content. Concretely, our system aims to:

- Detect and classify key slide regions (titles, body text, figures, etc.) and convert them into a structured layout representation.
- Generate a new background that follows this layout, keeps designated text regions smooth and uncluttered, and respects a chosen style or color palette.

2 Related Work

2.1 Slide Understanding and Computer Vision

Classifiers in computer vision analyze presentation slides by transforming visual contents into numerical representations that machine learning models can interpret. Initial steps include techniques such as image preprocessing/segmenetation and edge detection to help isolate elements like text blocks[6], images, charts, and icons[7]. Furthermore, feature extraction methods ranging from traditional descriptors (e.g., HOG or SIFT) to modern convolutional neural networks help capture patterns in shape, color, and layout. The extracted features are then fed into a classifier, which learns to associate visual patterns into categories, such as title, bullet points, diagrams, or charts. By combining these computer vision techniques, classifiers can accurately interpret and categorize slide elements, enabling automated slide analysis, indexing, or content understanding[5].

2.2 Diffusion Models and Generative Methods

Diffusion models have become a standard approach for high quality image synthesis by iteratively denoising random noises into realistic images. Early work operated directly in pixel space, but modern diffusion models

such as Stable Diffusion and its successor SDXL[17] move the process into a compressed latent space. This greatly reduces computation time and enables high resolution generation while preserving high image quality.[17]

Several works have explored adding stronger spatial control to diffusion models. ControlNet introduces an architecture that attaches a conditional branch to a frozen diffusion backbone, allowing the model to follow extra inputs such as edge maps, depth, segmentation, or human pose[4]. Our work is inspired by this, but instead of real-world conditioning signals, we use synthetic layout control maps and safe-zone masks that correspond to slide structure (title/body regions, etc.).

3 System Overview

Our system is composed of two primary pipelines that merge into a larger generation module:

1. **Visual and Text Analysis:** A slide understanding module that runs YOLO-OBB to detect structural components such as diagram nodes, images, and logos. We use EasyOCR[6] plus a Random Forest classifier to identify semantic text roles (e.g., Titles, Captions, Body text).
2. **Generative Background:** A diffusion-based module that takes the combined layout mask from the classifier pipeline, together with a user-provided text prompt (style and color palette). We use them to produce a new background that respects the slide safe zones and keeps text regions readable.

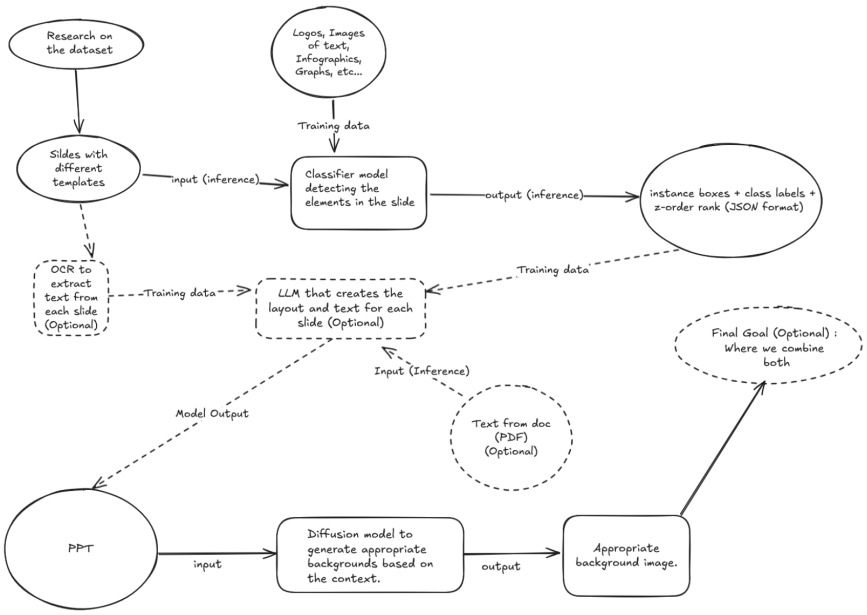


Figure 1: High-level system overview. The slide understanding pipeline converts a raw slide into a structured layout, then the diffusion model generates a layout aware background.

Our target users are instructors, students, and professionals who already know their content but lack the time or design skills to make clean and consistent slides. Typical use cases include cleaning up cluttered lecture slides, unifying the slide theme, and producing projector friendly backgrounds with better readability in text regions.

4 Method

4.1 Pipeline 1: Slide Understanding Module

The first pipeline of our architecture is responsible for parsing the unstructured slide image into a structured layout representation. This involves two parallel modules: visual detection and text analysis.

To support different levels of visual detail, we built two separate inference modes that mainly differ in the detection model they use. The slides mode runs the Logo and Image Classification model (Section 4.1.2) to find

logos and other embedded images. The diagrams mode instead uses the Visual Component Detection model (Section 4.1.1) to break down slides into their individual visual nodes. Both modes then rely on the same Charts and Shapes classifier (Section 4.1.3) and the shared text analysis pipeline (Section 4.1.4).

4.1.1 Visual Component Detection (YOLO-OBB)

To detect non-textual elements, we developed a custom YOLO-OBB (Oriented Bounding Box) model. Standard bounding boxes often fail on rotated diagrams or flowcharts. On the other hand, OBB allows the model to capture these oriented shapes accurately.

- **Target Class:** Initially, the model is focused on a single class to get every visual component: diagram_node, which includes all the visual parts of a slide, like the nodes of a diagram, shapes, images, etc.
- **Data Sources:** We used the AI2D[14] dataset to get high-quality bboxes for nodes in scientific diagrams, and also wrote a script to generate synthetic data. It places random elements like cloud icons, flowchart boxes, and stylized shapes onto slide-like backgrounds to help the model learn diagrams more related to the computer science field.

Split	Count	Percentage
Train	6320	80%
Validation	810	10%
Test	814	10%

- **Training:** We started by training on the mix of AI2D and the synthetic data. **mAP50:** 0.995, **mAP50-95:** 0.98. We then ran this model on actual lecture slides. The performance was not ideal.
- **Fine Tuning:** We manually corrected the detections on 100 lecture slides using the model trained above. Finetuning on this small "gold standard" set significantly improved the results on actual slides. **mAP50:** 0.95, **mAP50-95:** 0.87.

4.1.2 Logo and Image Classification (YOLO-OBB)

To distinguish between branding (Logos) and content (Images), we trained a secondary YOLO model.

- **Dataset:** We synthetically generated a dataset using logos from the Logo-2k+ dataset[7] and images from SlideVQA[5]. The generation script places 2 to 6 objects on random gradient backgrounds to prevent the model from learning shortcuts (e.g., assuming white backgrounds for logos).
- **Training:** The model achieved a **mAP50** of 0.998 on the test set, differentiating between logos and images.

4.1.3 Charts and Shapes Classification

To classify types of charts and shapes in slides, we trained a YOLO model with single and multi-class scenarios to better distinguish between different elements.

- **Synthetic Dataset:** The SlideVQA dataset[5] that was used in visual component detection contains three broad categories: Diagram, Image, and Figure. To address this limitation, we generated a custom dataset using Matplotlib with 23 different classes to provide a better labeled dataset to train the custom YOLO model.
- **Training Model:** For each class, we generated 500 images, ensuring a sufficient amount and variation of dataset for training. And to more effectively train the classifier for multi-class scenarios, we created 2,000 images that contain 3 to 5 randomly selected classes. This approach helps the model learn to detect multiple objects within a single slide. The model was trained through 30 epochs to ensure sufficient training.

4.1.4 Text Extraction and Semantic Classification

Parallel to the visual detection, we implemented a text analysis pipeline to extract not just the content, but the semantic role of text elements.

1. Extraction Phase: We utilized EasyOCR[6] to detect text regions and extract raw string content along with their bounding box coordinates (x, y, w, h).
2. Feature Engineering: Since raw coordinates are insufficient for classification across different templates, we implemented a feature extractor that generates a 19-dimensional feature vector for every detected text box. Key features include:
   - Geometric: relative_width and relative_area (to identify titles).
   - Positional: relative_y (to identify Headers/Footers).
   - Visual: edge_density and mean_brightness (to distinguish text on complex backgrounds).
3. Classification Model: We trained a Random Forest Classifier (200 estimators, max depth 15) on the SlideVQA dataset. The model classifies text into five categories: Title, Caption, Obj-text, Page-text, and Other-text.

4.2 Pipeline 2: Generative Module

The generative pipeline takes the structured layout from the classification pipeline, together with a user-provided text prompt (style and color palette) and uses them to guide a Stable Diffusion XL (SDXL) model to produce a new background that respects the slide safe zones. In our current prototype, the prompt and palette are chosen manually. The original design included a LLM to suggest them automatically, which we leave as future work. We treat SDXL as a general image prior and adapt it with Low-Rank Adaptation (LoRA) on the UNet attention layers. We then freeze the original weights and train only small low-rank updates, which makes specialization to slide-safe backgrounds feasible on a single GPU.

For training data, we sample roughly 5,000 images from the Describable Textures Dataset (DTD)[16] and pair each with a caption that includes a custom Slidesafe token indicating slide background with no overlaid text or logos. In parallel, we use a synthetic dataset about 5,000 layouts generated by our pipeline. Each layout provides (i) a layout control map encoding where titles, body text, images, and logos should appear, and (ii) a safe-zone mask marking regions intended for text. During training, this mask is used to locally smooth the corresponding regions in the DTD images, nudging the model to keep masked areas visually calm while allowing richer structure elsewhere.

We fine-tune SDXL + LoRA at 512×512 resolution in Google Colab using the standard diffusion objective: images are encoded into latents, noise is added at random timesteps, and the LoRA-augmented UNet learns to predict the noise. At inference time, we attach a pretrained ControlNet to SDXL and feed it a rendered control image derived from the layout control map, while the text prompt specifies the desired style and palette. We compare two variants under identical seeds and sampling settings: a baseline SDXL model that uses only text prompts, and our layout-aware model (SDXL + LoRA + ControlNet + safe-zone smoothing), whose outputs are evaluated in Section 5.

5 Experiments and Evaluation

5.1 Pipeline 1: Slide Component Classifier Performance

5.1.1 Visual Component Detection (YOLO-OBB)

We evaluated the Component Detection model using Mean Average Precision (mAP) at IoU thresholds of 0.50 and 0.50-0.95.

1. Initial Training: **mAP50:** 0.995 and **mAP50-95:** 0.98
2. Fine-Tuning on real slides: **mAP50:** 0.95 and **mAP50-95:** 0.87

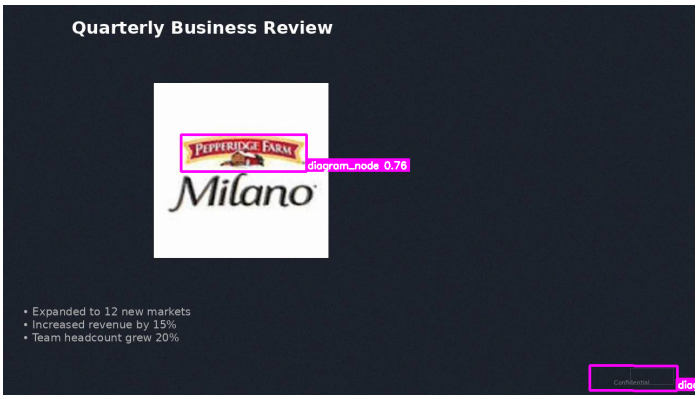


Figure 2: Model correctly ignores text and only focuses on the visual parts

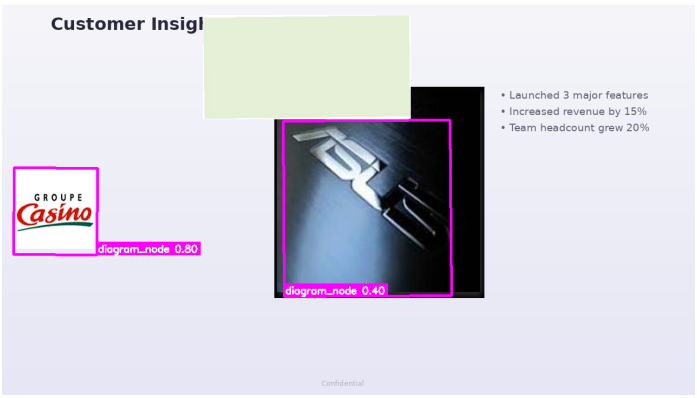


Figure 3: Model only detects visual components

5.1.2 Logo/Image and Charts/Shapes Detection

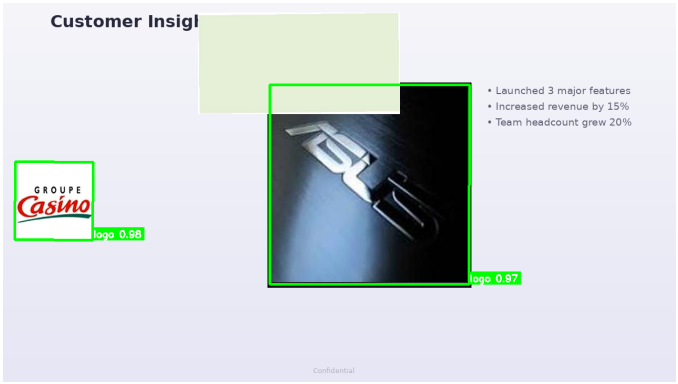


Figure 4: Detections using the Logo/Image model

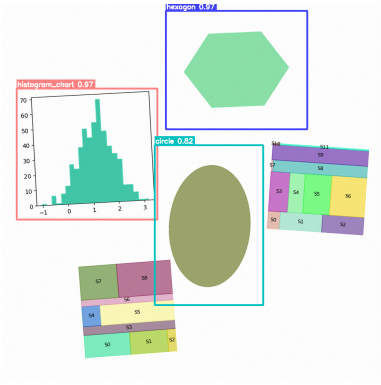


Figure 5: Detections using the Charts/Shapes model

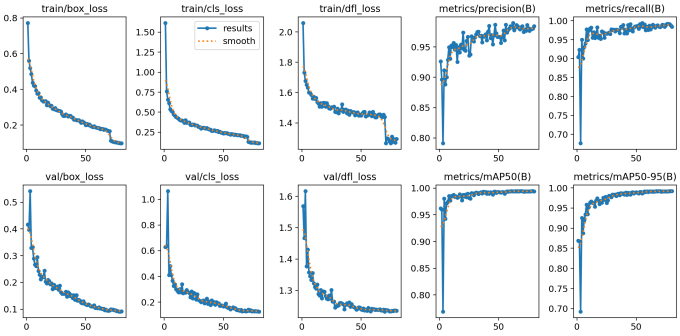


Figure 6: Training and validation loss curves for Logo/Image Classification

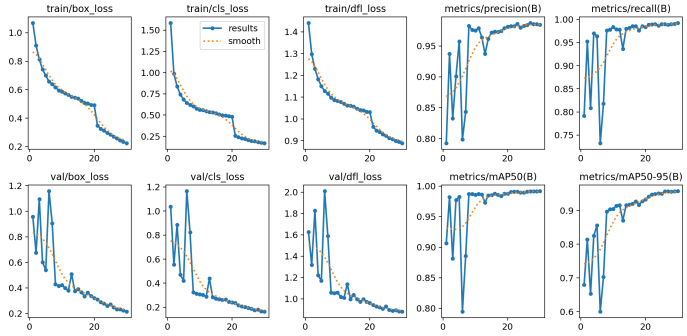


Figure 7: Training and validation loss curves for Charts/Shapes Classification

5.1.3 Text Classification Results

We evaluated the Random Forest Text Classifier on a stratified test set of 14,104 samples from SlideVQA. The model achieved an overall Test Accuracy of 81.91%.

1. Class-Wise Performance: As shown in Table 1, the model demonstrates high variance in performance depending on the semantic distinctiveness of the text role.

Class	Precision	Recall	F1-Score	Support
Title	0.87	0.88	0.88	1852
Obj-text	0.84	0.95	0.89	7825
Other-text	0.87	0.68	0.76	973
Page-text	0.69	0.67	0.68	2607
Caption	0.62	0.08	0.14	847

Table 1: Performance Metrics for Text Classification.

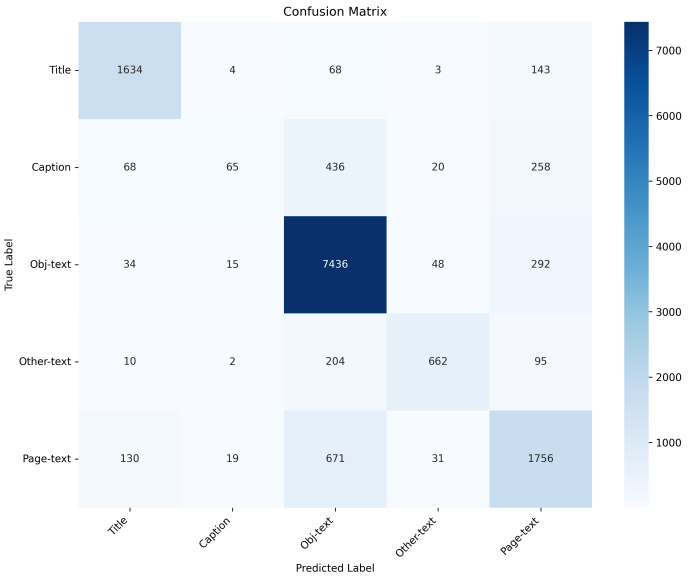


Figure 8: Confusion Matrix for Text Classification.

2. Error Analysis: The Confusion Matrix (Figure 8) reveals the specific limitations of using geometric features alone:
   - Success: Title and Obj-text have high recall (0.88 and 0.95 respectively). Their spatial signals are consistent, showing the titles at the top and object text inside diagrams.
   - Failure Mode: Caption performance is poor (Recall: 0.08). The model frequently misclassified captions as Obj-text. This is due to spatial ambiguity: captions often appear immediately below or adjacent to diagrams, sharing similar font sizes and bbox dimensions as the diagram labels themselves. Without semantic understanding of the string content, they are geometrically indistinguishable.

5.2 Pipeline 2: Diffusion Model Results

5.2.1 Quantitative Evaluation

We evaluate the diffusion component by comparing a baseline SDXL model (prompt only) to our layout-aware model (SDXL + LoRA + ControlNet + safe-zone smoothing) on 75 matched A/B image pairs. For each pair, we regenerate the same synthetic layout and safe-zone mask and compute layout-aware readability metrics.

Table 2 summarizes the key results. The layout aware model reduces the standard deviation of grayscale intensities in masked regions (safe_std) by roughly 28% on average. This indicates smoother, less cluttered backgrounds for the masked area. It also improves WCAG-style contrast pass rates, especially inside the safe zones, while slightly increasing a derived smoothness score.

Model	Safe-zone std (↓)	WCAG safe-zone (↑)	Smoothness (↑)
Baseline SDXL	0.1382 ± 0.0317	0.678	0.447
SDXL + LoRA (ours)	0.0967 ± 0.0409	0.719	0.613

Table 2: Diffusion model metrics (baseline SDXL vs layout-aware SDXL + LoRA). Values shown as mean ± std over 75 A/B pairs.

5.2.2 Qualitative Examples

Figure 9 shows a representative A/B grid across all five style presets (default, academic, noisy, gradient, photo). Each row contains the control / layout mask, the baseline SDXL output, the LoRA output, and the LoRA output with the mask overlay. Across presets, the LoRA model consistently flattens the title and body bands while keeping more variation in background only areas.

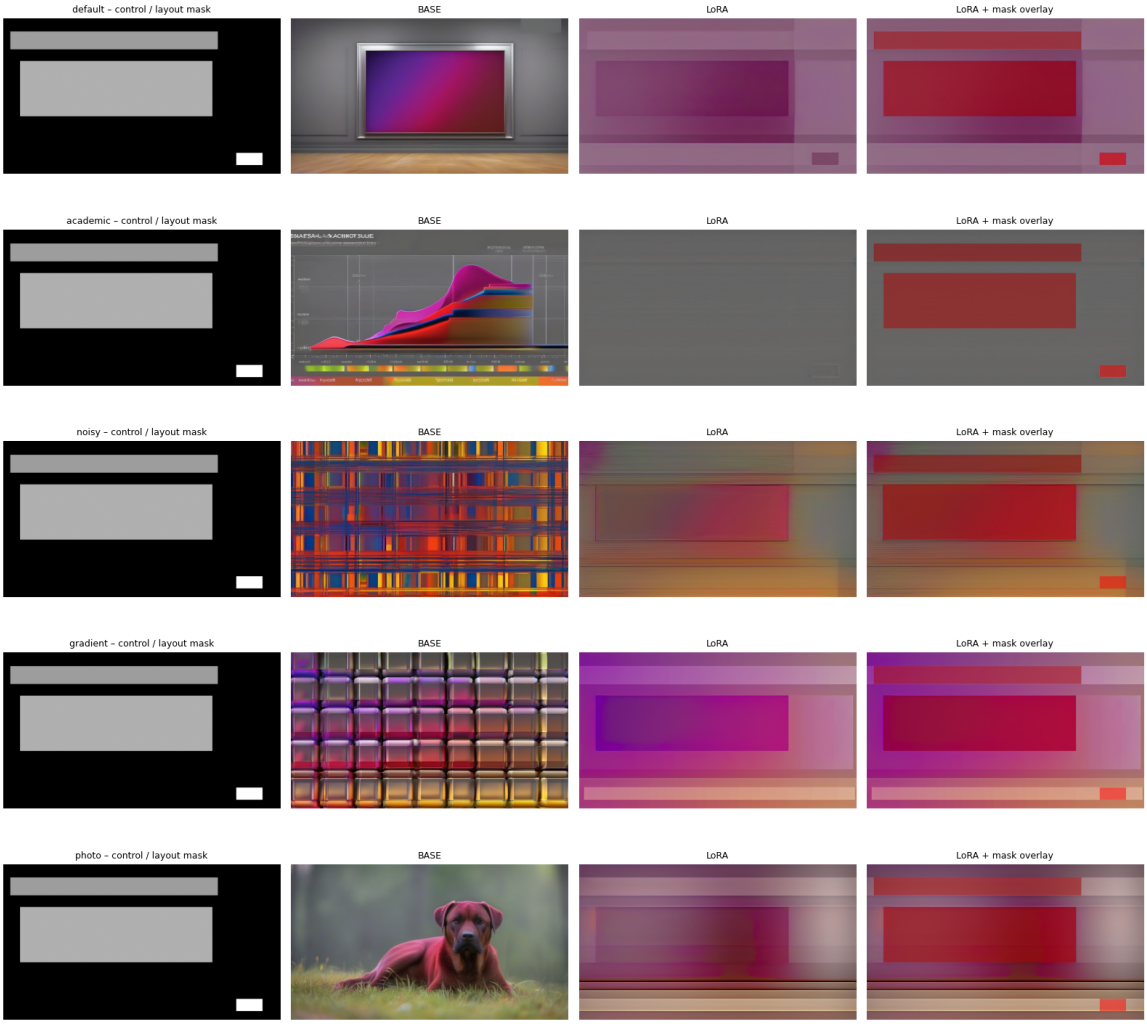


Figure 9: A/B grid for five style presets. From left to right: control / layout mask, baseline SDXL, SDXL + LoRA, and SDXL + LoRA with mask overlay. LoRA produces smoother regions under the title/body boxes while preserving overall style.

To zoom in on one case, Figure 10 shows a photo preset example. The baseline SDXL model produces a regular photographic image, which is visually striking but not usable as a slide background because the subject competes with text. After Slidesafe fine-tuning and layout conditioning, the same prompt and seed produce an abstract background with a calm title bar and a large, smooth body region in the center, better suited for overlaid text.

(a) Baseline SDXL

(b) SDXL + LoRA (layout aware)

Figure 10: Zoomed photo example. With the same prompt and seed, the layout aware model transforms a generic photographic output into a slide-safe background with clearly defined, smooth bands for title and body text.

6 Use of AI Tools

We used OpenAI's ChatGPT, Google's Gemini, and/or other LLMs as a support tool throughout this project, both during system development and while preparing the written materials. Their usage was limited to brainstorming, clarifying concepts, proofreading, and reinforcing our understanding of the topics. It was used to:

- Brainstorm and refine design decisions for the classifier and diffusion modules, including discussing alternative architectures, evaluation strategies, and ablation ideas,
- Assist with debugging by helping interpret error messages, suggest possible failure modes, and propose diagnostic checks,
- Support the proofreading of our documentation and written deliverables (including this report), mainly by suggesting clearer phrasing, alternative structures, and ways to improve readability.

All core implementation, experimental design, and final conclusions were authored, executed, and validated by the team.

7 Limitations and Future Work

7.1 Current Limitations

A significant limitation of our current pipeline lies in its generalization to complex, real-world slide layouts. The chart and shapes classifier, trained entirely on synthetic data, biases heavily toward the clean uniformity of tools like Matplotlib and may struggle with the diverse designs and noisy backgrounds in actual presentations. Similarly, the text classification module faces challenges with multimodal ambiguity. Relying strictly on geometric features proves that it is insufficient for distinguishing captions from diagram text, which share similar spatial characteristics but differ fundamentally in semantic role.

Our diffusion module has a few key limitations. First, the LoRA is trained on DTD textures and synthetic layouts rather than real slide backgrounds. This causes a domain gap and some outputs still look more like textures than polished slides. Second, we only reuse a pretrained ControlNet instead of fine-tuning it on our layout control maps. This means the model does not always perfectly respect the intended structure and some texture can still bleed into text regions. Third, the Slidesafe LoRA can be too strong in some presets (especially photo and noisy). This means that the model heavily flattens the image, improving readability but also removing much of the visual creativity of the base SDXL model.

7.2 Future Work

To address these limitations, future work will focus on enhancing data diversity and model integration. For graph classification, we aim to incorporate real-world examples. The goal is to reduce the synthetic bias of our current Matplotlib-based training set, thereby improving the robustness of our graph classifier across varied slide designs. Simultaneously, we plan to deepen the integration between the visual and text pipelines by feeding YOLO visual features directly into the text classifier. The could explicitly allow the model to learn contextual cues such as a text block being near a diagram to resolve the ambiguity between captions and object labels.

For the diffusion model, a natural next step is to fine-tune a layout-aware ControlNet on our synthetic control maps and, if possible, on real layouts derived from slides. We would also like to train on a curated set of real slide backgrounds or de-textified slides to reduce the domain gap. On the modeling side, exploring weaker or style-dependent LoRA settings could better balance readability and creativity. Longer term, we aim to move from background generation to full slide editing, where the model inpaints only the background behind detected text boxes, and to complement our automatic metrics with small user studies on perceived readability.

More broadly, the classifier and diffusion components are intended to be only two pieces of a larger slide-generation ecosystem that includes LLMs, agents, and higher-level planning modules. Building and integrating these additional components, such as automatic content analysis, prompt and palette suggestion, and end-to-end slide authoring flows, remains substantial future work beyond the current prototype.

8 Individual Reflections

8.1 Student 1: Anirud Mohan

My primary focus in this project was developing the Text Extracting pipeline and integrating it with the visual streams. I began with a naive assumption since I thought simple bounding box coordinates would be sufficient to distinguish text roles. However, early experiments yielded poor results, forcing me to pivot. I quickly realized that robust classification required a much richer feature set, leading me to engineer 19 distinct features, including visual density and relative area which significantly boosted accuracy.

On the modeling side, I experimented with a Support Vector Machine (SVM) but found that a Random Forest Classifier not only outperformed it in overall accuracy but also offered faster training times, which was crucial given the size of our dataset. Implementing the full pipeline was a significant coding challenge, particularly harmonizing EasyOCR's output with my custom predictor. I leveraged AI coding assistants (GitHub Copilot) to help debug complex integration errors and optimize the feature extraction loops for speed.

Finally, working with the SlideVQA dataset highlighted the messy reality of real-world data. The persistent confusion between Captions and Diagram Labels was a humbling reminder that some visual ambiguity is hard to resolve with tabular features alone. If I were to extend this work, I would likely train a dedicated YOLO model specifically to detect Caption regions as a visual object, rather than relying solely on the text classifier to infer it.

8.2 Student 2: Javad Baghirov

8.2.1 Visual Element detection model

When I first tried building the diagram node detector, my plan was just to gather existing datasets, combine them, and train YOLO-OBB. Pretty quickly I realized there wasn't really any dataset that matched what I actually needed. The AI2D dataset helped a bit, but models trained on it didn't transfer well to real lecture slides at all. The diagrams in AI2D look nothing like CS slides, so the model completely struggled.

To fix this, I ended up creating my own synthetic diagram generator that produced CS-style diagrams. I added as much variation as I could, like different backgrounds, shapes, scales, and colors. This definitely improved things, but I also learned that synthetic data by itself isn't enough. The model still failed on real slides because it had never actually seen them.

What worked best was using the partially trained model to auto-label real slides, then going in and correcting the predictions by hand. Fine-tuning on these corrected real examples made the biggest difference. The main thing I learned here is that making synthetic data realistic is quite challenging, and a small amount of high-quality real data can fix a lot of the generalization issues.

8.2.2 Image and Logo Detection Model

For the logo versus image classification task, I originally thought this would be pretty easy, but it turned out that the model needed a carefully designed dataset to learn the difference properly. Since I couldn't find a dataset that matched how logos and images actually appear on slides, I ended up generating the entire dataset myself. I used logos from the Logo-2K+ dataset and regular images from SlideVQA, then built a script that placed 2 to 6 objects on gradient backgrounds with random titles and colors. This was important because I learned that if the backgrounds stay the same, the model will start cheating by using the background pattern instead of the actual object features.

The key challenge here was preventing the model from relying on shortcuts, like associating logos with white backgrounds. By randomizing backgrounds and adding text, I forced the model to focus on the actual shapes and textures of logos versus images. This made the model much more robust on real slides.

8.3 Student 3: Hsing-Hao Wang

Initially, I planned to use a convolutional neural network to perform the classification task on the SlideVQA dataset. However, I found this approach to be too time-consuming and inefficient for this classification problem. In addition, the SlideVQA dataset contains only annotated slides of diagrams, images, and figures, which did not align well with my goal of differentiating between specific types of graphs, charts, and shapes. After doing

research, this led me to adopt a YOLO-based model, which is better suited for detecting and classifying objects. Since my objective was to differentiate chart types (such as bar charts and line charts), which is categorize as diagrams or images in the SlideVQA dataset. It made more sense to generate a synthetic dataset tailored to these categories. The resulting custom YOLO model performed well and was able to accurately classify the different chart types. One improvement I would pursue is to include actual chart and shape examples into my synthetic dataset to enhance model robustness. Currently, the dataset consists entirely of synthetic images, which may limit performance when face charts that appear differently from those produced by Matplotlib.

8.4 Student 4: William C. Loe

Coming into this project, I mainly wanted to learn how to take a large diffusion model and bend it toward a very specific, practical task. I ended up owning essentially the entire generative side of the system: curating the texture and synthetic layout datasets, designing the layout/control-mask format, building the SDXL+LoRA training pipeline, and setting up the A/B evaluation and visualizations. One teammate was originally supposed to split the diffusion work with me, but their availability changed, so I took full ownership of this stream while the others focused more on the classifier pipeline.

Technically, the project pushed me much deeper into the "production" side of diffusion than previous homeworks. I had to figure out how to run SDXL and LoRA reliably on Colab, keep track of runs with different seeds and style presets, and debug behavior like the LoRA becoming too strong and flattening out most of the base model's creativity. Designing the Slidesafe token, safe-zone smoothing, and layout control maps forced me to think of the model as something I could steer through data and conditioning, not just a black-box image generator. I also realized how non-trivial evaluation is: defining metrics like safe-zone variance and simple WCAG-style checks made me think carefully about what "readability" actually means in a measurable way.

On the non-technical side, I learned a lot about planning and division of work. Our initial diagram was quite ambitious, with an LLM generating layouts and prompts and a fully automatic end-to-end pipeline. Over time, I had to prioritize getting a solid classifier + diffusion path working with manually specified prompts and palettes, and treat the LLM and full editing loop as future extensions instead of core requirements. That shift taught me to be more realistic about what can be built within a course project, while still keeping the long-term vision in mind.

Overall, I'm proud that we got a layout-aware diffusion system working and that we can show both quantitative improvements and clear A/B visual examples. The project gave me hands-on experience with adapting a large diffusion model to a niche use case, coordinating with a separate classifier pipeline, and thinking about usability (readable, projector-friendly slides) rather than just pretty images.

9 Bonus Tasks

9.1 Bonus Task - 2: User Study or Survey

9.1.1 Small-Scale User Study with Qualitative Metrics

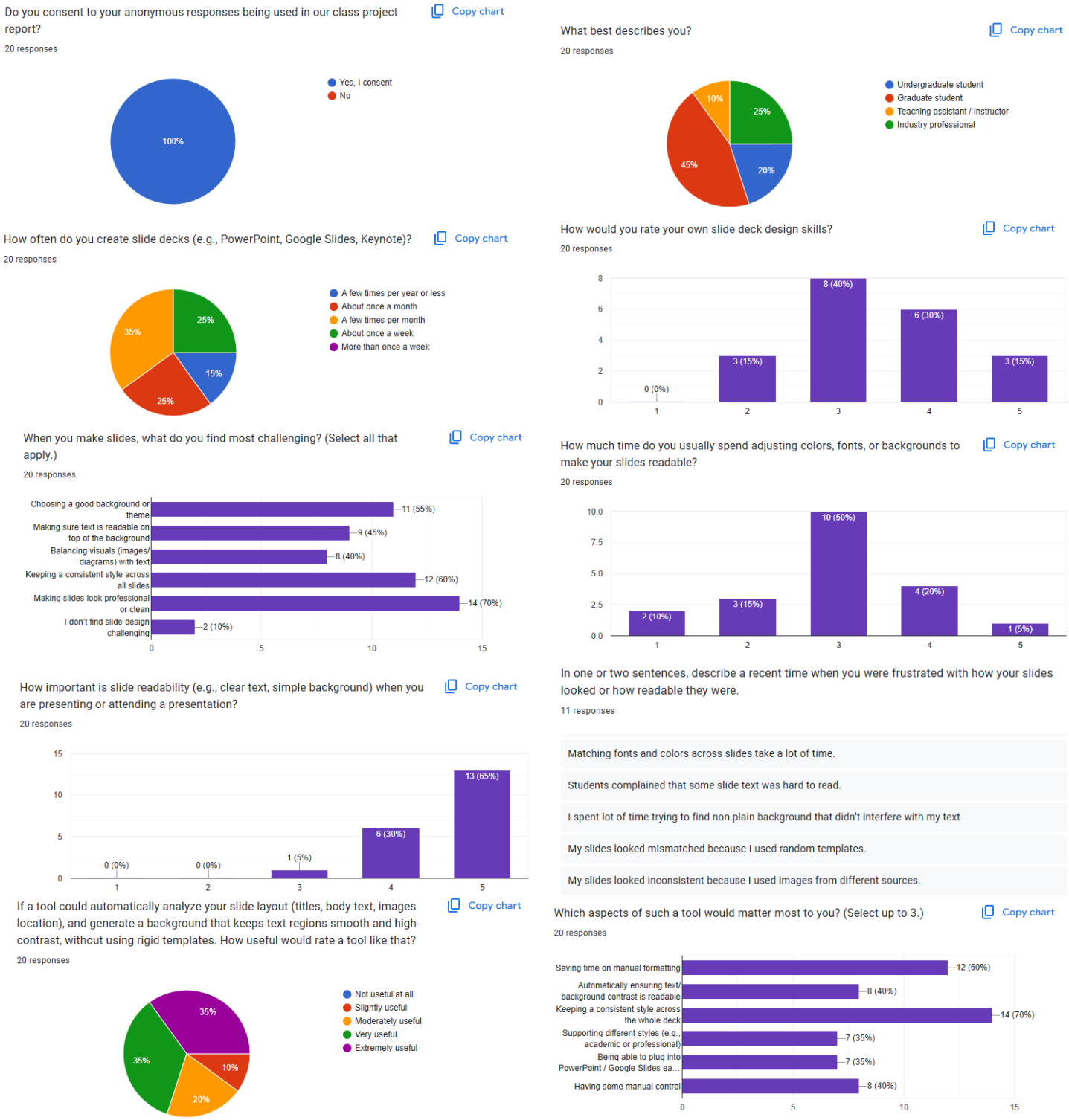


Figure 11: Summary of user survey results

The survey gathered 20 responses from different graduate students, industry professionals, undergraduate students, and instructors. Most participants create presentation slide regularly and rated their design skills around the middle of the scale. The most common challenges among participants include making the slides look professional

(70%), keeping a consistent style (60%), choosing an appropriate background or theme (55%), and ensuring readable text/background contrast (45%). Many participants reported frustrations such as dark backgrounds reduces readability, mismatched fonts and colors, and difficulty finding non-distracting backgrounds. Participants typically spend an average amount of time adjusting colors, fonts, and backgrounds to improve readability of the slide, which almost all of them stated that slide readability is very important for a presentation. When participants are asked about a tool that could automatically analyze slide layout and generate high-contrast, readable backgrounds without relying on rigid templates, 70% rated it as very or extremely useful. The most valued features of our tool include maintaining consistent style throughout a presentation, saving time on manual formatting, ensuring readable contrast, and integrating easily with PowerPoint or Google Slides. Overall, the survey results highlight a strong need for a generating system to improve slide readability and consistency.

9.1.2 Reflection on Social Impact

For social impact, the slide template generator allows users without design backgrounds to create polished and professional presentations. At the same time, this can enhance accessibility to high quality communication tools in educational and professional settings. Therefore, the generator maintains diversity and flexibility in generated outputs to support a wide range of circumstance and stylistic preferences.

9.2 Bonus Task - 3: Ethical or Social Considerations

Our system operates on slide images that may contain sensitive or proprietary information (e.g., student grades, internal company data, logos). A first ethical concern is privacy and confidentiality. If slides are uploaded to a cloud service, there is a risk of exposing content that was never meant to leave a classroom or organization. To mitigate this, the system should (i) allow fully local/on-device processing where possible, (ii) avoid storing raw slides or OCR text beyond the current session, and (iii) give users clear controls over what is saved, logged, or used for future model training.

A second issue is misinterpretation or distortion of content. Our goal is to redesign backgrounds, but an over aggressive pipeline could move or hide important elements, making a slide misleading (for example, visually de-emphasizing a warning or changing the apparent hierarchy of information). To address this, our design keeps the generator focused on background-only changes, and we explicitly separate "content layers" (titles, body text, diagrams) from backgrounds. A practical mitigation is to present the redesigned slide as a preview and require explicit user confirmation before export, keeping a clear human-in-the-loop.

Third, there is a risk of bias and accessibility gaps. Our training data (textures and synthetic layouts) may reflect particular aesthetic choices and assume left-to-right, top-heavy layouts that do not match all cultures, disciplines, or accessibility needs. In addition, our current readability metrics only approximate WCAG-style contrast and ignore font, colorblindness, and screen/projector settings. To mitigate this, future versions should (i) incorporate more diverse slide examples, including non-English and right-to-left layouts, (ii) expose user-adjustable accessibility presets (e.g., high-contrast mode), and (iii) complement automatic metrics with user studies that include people with different visual and cognitive needs.

Finally, because the system can make it easier to produce polished slides at scale, it could be used to create persuasive but low-effort content (e.g., spammy or misleading decks). While we cannot fully prevent this, we can reduce harm by positioning the tool clearly as a design assistant, not an "auto-content generator," and by avoiding features that fabricate textual content or data. In summary, our design choices and mitigation aim to keep users in control, protect sensitive material, and prioritize readability and accessibility over purely aesthetic optimization.

9.3 Bonus Task 4: Data Collection and Enhancement

9.3.1 Original Data and Annotation

The public datasets we started with were helpful, but they did not fully match the style of real lecture slides. To close this gap, we created a small but high-quality dataset ourselves.

- **Manual Annotation:** We collected around 100 real lecture slides from our previous courses and created our own annotated dataset. To speed up the process, we first ran a partially trained model (trained on

synthetic data and public datasets) on these slides, and then manually corrected any mistakes it made. This allowed the capture of the rotated images, illustrations, and diagram components more accurately .

- **Why This Was Needed:** These hand-annotated slides served as a "gold standard" and were especially important during the fine-tuning stage. This dataset helped connect the cleaner AI2D diagrams with the much messier layout of actual course material.

9.3.2 Simulation and Augmentation Strategy

- **Synthetic Diagram Generation:** A Python script that generates diagrams by randomly placing icons (clouds, databases, rhombuses, etc.) and connecting them with arrows.
- **Slide Composition for Logos vs. Images:** A separate generator that places images or logos on different backgrounds, adds text, and varies lighting and colors. This produced a high-variance dataset without needing to label thousands of samples.

9.3.3 Challenges and Solutions

One major issue throughout the project was domain shift. Models pre-trained on AI2D worked well on simple white diagrams but struggled on real lecture slides.

- **Issue:** The model started relying on background cues instead of the object itself, for example treating "white space" as a signal for logos.
- **Fix:** The synthetic augmentations (different gradient backgrounds, random text) forced the model to ignore the background and actually pay attention to shapes, edges, and textures. After this change, the model performed noticeably better on actual slides.

9.3.4 Dataset Distinctness

Each model was trained on its own independent dataset.

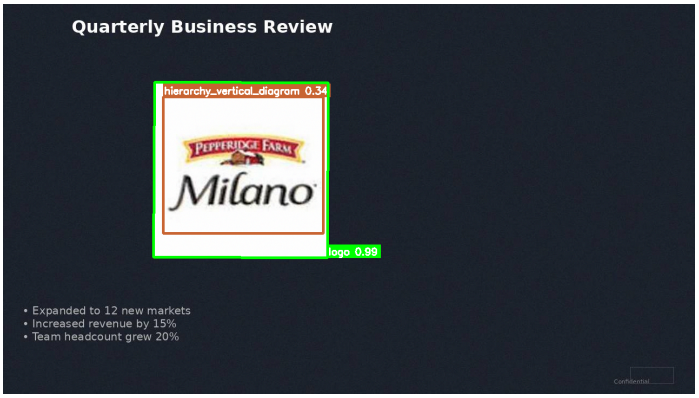
- **Task 1 — Component Detection:** Trained using AI2D plus the synthetic flowchart diagrams. Fine tuned on real slides. This dataset focuses only on structural elements (nodes, images, blocks).
- **Task 4 — Logo Classification:** Trained using Logo-2K+ and SlideVQA, along with my synthetic slide generator.

9.4 Bonus Task - 5: Cross-Modal Integration

To fulfill the requirement of cross-modal integration, we extended our system to fuse Textual Data (via the OCR + Classifier pipeline) with the Visual Data (YOLO-based object detection). We implemented a modular control flag within the main inference pipeline to treat text as an independent modality.

9.4.1 Value Assessment

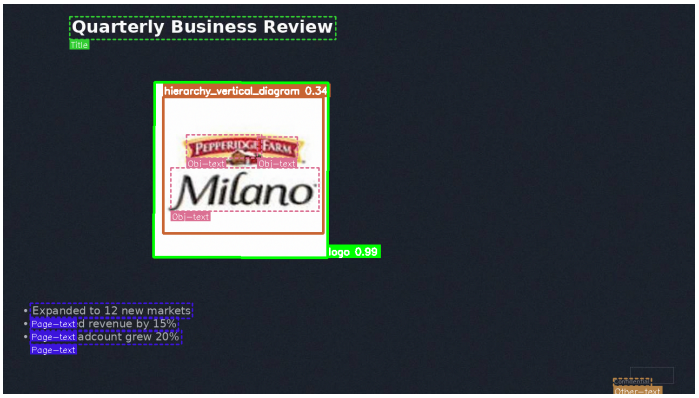
We compared the system's output on identical slides with the text modality disabled versus enabled. The integration provides both Qualitative (Semantic) and Quantitative (Detection Count) improvements.



(a) **Visual Only (Image):** The model detects structural elements like Diagram Nodes but lacks semantic labels.


```
[
  {
    "DETECTION_STATISTICS": {
      "mode": "vision_only",
      "total_slides_processed": 6,
      "total_detections": 33,
      "detections_per_slide": {
        "min": 2,
        "max": 10,
        "average": 5.5
      },
      "breakdown": {
        "graphs": 18,
        "images_logos": 15
      }
    }
```


(b) **Visual Only (Stats):** Detection count is limited to just graphic elements (33 detections).



(c) **Multimodal (Image):** The system overlays Title and Caption labels, anchoring the diagram to a topic.


```
[
  {
    "DETECTION_STATISTICS": {
      "mode": "multimodal",
      "total_slides_processed": 6,
      "total_detections": 117,
      "detections_per_slide": {
        "min": 10,
        "max": 57,
        "average": 19.5
      },
      "breakdown": {
        "graphs": 18,
        "images_logos": 15,
        "text": 84
      }
```


(d) **Multimodal (Stats):** Total detections jump to 117, as the text stream captures 84 additional semantic elements.

Figure 12: Comparative analysis of system outputs. The top row (Vision Only) shows limited context. The bottom row (Multimodal) demonstrates a massive increase in information density.

As shown in Figure 12, the integration adds significant value by increasing the total detections per batch from **33 to 117** and providing semantic interpretability (e.g., identifying the Results title) that visual detection alone lacks.

References

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[2] D. Podell et al. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*, 2021.

[4] L. Zhang, A. Rao, and M. Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[5] R. Tanaka, K. Nishida, and S. Yoshida. SlideVQA: A Dataset for Document Visual Question Answering on Multiple Images. *arXiv preprint arXiv:2301.04883*, 2023.

[6] JaidedAI. EasyOCR: Ready-to-use OCR with 80+ supported languages. *GitHub repository*, `https://github.com/JaidedAI/EasyOCR`, 2020.

[7] J. Wang, W. Min, S. Hou, S. Ma, Y. Zheng, H. Wang, and S. Jiang, "Logo-2K+: A Large-Scale Logo Dataset for Scalable Logo Classification," *arXiv preprint* arXiv:1911.07924, 2019.

[8] W. Tang, J. Xiao, W. Jiang, X. Xiao, Y. Wang, X. Tang, Q. Li, Y. Ma, J. Liu, S. Tang, and M. R. Lyu, "SlideCoder: Layout-aware RAG-enhanced Hierarchical Slide Generation from Design," in *Proc. EMNLP*, 2025.

[9] T. Shirakawa and S. Uchida, "NoiseCollage: A Layout-Aware Text-to-Image Diffusion Model Based on Noise Cropping and Merging," in *Proc. CVPR*, 2024.

[10] Q. Zhangli, J. Jiang, D. Liu, L. Yu, X. Dai, A. Ramchandani, G. Pang, D. N. Metaxas, and P. Krishnan, "Layout-Agnostic Scene Text Image Synthesis with Diffusion Models," in *Proc. CVPR*, 2024.

[11] H. Guo, Z. Chen, Z. Li, and J. Zhu, "ContentDM: A Layout Diffusion Model for Content-Aware Layout Generation," *IEEE Trans. Artificial Intelligence*, 2025.

[12] W3C, "Web Content Accessibility Guidelines (WCAG) 2.2," World Wide Web Consortium Recommendation, 2024. Available at `https://www.w3.org/TR/WCAG22/`.

[13] K. Sherwin, "Ensure High Contrast for Text Over Images," Nielsen Norman Group, Oct. 2015. Available at `https://www.nngroup.com/articles/text-over-images/`.

[14] K. Ahmed, X. Li, and B. Van Merrienboer, "The AI2 Diagrams Dataset: A Annotated Corpus for Diagram Understanding," Allen Institute for Artificial Intelligence (AI2), 2016. Available at `https://allenai.org/data/diagrams`.

[15] OpenAI. ChatGPT (GPT-5.1 Thinking), accessed 2025. Available at `https://chat.openai.com`.

[16] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing Textures in the Wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[17] D. Podell, T. Müller, A. Serrano, T. Hu, A. Gharbi, A. Esteves, S. Kautz, B. Poole, and B. Ommer, "SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis," *arXiv preprint arXiv:2307.01952*, 2023.