AI-Generated Lecture Slides for Improving Slide Element Detection and Retrieval

Suyash Maniyar*[1][0009−0000−5882−4377], Vishvesh Trivedi*[2][0009−0004−5043−6766], Ajoy Mondal[3][0000−0002−4808−8860], Anand Mishra[1][0000−0002−7806−2557], and C. V. Jawahar[3][0000−0001−6767−7057]

[1] CSE, Indian Institute of Technology, Jodhpur, India
suyash.1@alumni.iitj.ac.in, mishra@iitj.ac.in
[2] Sardar Vallabhbhai National Institute of Technology, Surat, India
u20cs130@coed.svnit.ac.in
[3] CVIT, International Institute of Information Technology, Hyderabad, India
{ajoy.mondal, jawahar}@iiit.ac.in

Abstract. Lecture slide element detection and retrieval, key tasks in lecture slide understanding, have gained significant attention in the multimodal research community. However, annotating large volumes of lecture slides for supervised training is labor-intensive and domain-specific. To address this, we propose a large language model (LLM)-guided Synthetic Lecture Slide Generation (**SynLecSlideGen**) pipeline that produces high-quality, coherent slides, named the **SynSlide** dataset, closely resembling real lecture slides. We also create an evaluation benchmark **RealSlide** by manually annotating 1050 real slides curated from lecture presentation decks. To evaluate the effectiveness of **SynSlide** dataset, we perform few-shot transfer learning on real slides using models pretrained on our synthetically generated slides. Experimental results show that few-shot transfer learning outperforms training only on the real dataset, especially in low-resource settings. It demonstrates that synthetic slides can be a valuable pre-training resource in labeled data-scarce real-world scenarios. Code and resources are available at our project page: https://synslidegen.github.io/.

Keywords: Lecture slide understanding · slide element detection · text-based slide retrieval · synthetic slide generation · few-shot transfer learning · large language model.

1 Introduction

Presentations are a ubiquitous form of business document used across various domains. Industry sectors such as business, healthcare, law, and engineering rely heavily on presentations as a primary medium for information exchange. Automatically parsing lecture slides and extracting information has many potential applications; examples include automatic summarization, intelligent search, and

* These authors contributed equally.

enables AI-powered educational assistants. Despite their widespread applications, automating the understanding and analysis of slide content remains a significant challenge primarily due to the absence of a large-scale, densely annotated dataset specifically designed for lecture slide images. Current Vision-Language Models (VLMs) for document understanding [32,11,1,6] heavily depend on large-scale annotated datasets to adapt pre-trained representations to novel domains. For example, significant improvements in Document Layout Analysis for scientific papers have been achieved through fine-tuning extensive datasets such as PubLayNet [36] and DocBank [17]. However, obtaining similar high-quality annotations for lecture slide images requires substantial manual labor due to their inherently complex layouts and high stylistic variability.

Although previous efforts in presentation slide understanding have achieved incremental progress in tasks such as *Slide Image Segmentation* [7,8], *Slide Image Retrieval* [15,12], and *SlideVQA* [31], the persistent challenge of manual data annotation remains a significant bottleneck. Inspired by the growing trend of leveraging synthetic data to enhance training across various domains, we introduce **SynLecSlideGen** — an LLM-guided open-source tool to generate coherent, realistic synthetic lecture slide images and corresponding automatic annotations. We evaluate our generated slide dataset **SynSlide** on two primary tasks of slide understanding – *Slide Element Detection (SED)* and *Text-based Slide Image Retrieval (TSIR)*. A challenging test set of 1,050 real lecture slide images, manually annotated with dense slide element detection labels and detailed textual summaries, assesses performance on both tasks. To assess the effectiveness of the **SynSlide**, we conduct few-shot transfer learning on real slides using models pre-trained on synthetic slides. Results show that this approach outperforms training solely on real data, highlighting the value of synthetic slides as a pre-training resource in low-annotation scenarios. Pre-training on synthetic slides before fine-tuning with just 50 real images enhances SED performance, achieving a 9.7% mAP boost on YOLOV9, with significant improvements in low-resource classes such as code snippets (+32.5% mAP) and natural images (+20.2% mAP). In the TSIR task, synthetic slide images improved R@1 accuracy by 3% with the CLIP model, showcasing its effectiveness in handling high intra-class variance and rare slide elements. These experiments highlight the potential of synthetic data to enhance performance in low-resource slide image scenarios, providing a scalable solution for understanding lecture presentations.

We summarize our contributions as follows.

- **SynLecSlideGen:** An LLM-based open-source pipeline for generating realistic, coherent, copyright-free lecture slides with automatic annotations, used to create the SynSlide dataset for slide understanding.
- **Benchmark Dataset:** A curated RealSlide dataset of 1,050 real lecture slides with manual annotations for slide element detection and query-based retrieval.
- **Comprehensive Evaluation:** We conduct extensive experiments to assess the performance of synthetic slides on two main tasks related to document understanding: slide element detection and query-based slide image retrieval.

Experiments show that pre-training on synthetic slides improves few-shot transfer learning on real slides, highlighting their value in low-annotation settings.

2 Related Work

2.1 Slide Image Understanding

Slide image understanding is a multidisciplinary field that includes tasks such as slide segmentation & narration and multi-modal retrieval.

Slide Segmentation & Narration: SPaSe [7] provides a multi-label slide segmentation dataset with 2,000 pixel-annotated slides from SlideShare1M [2], addressing overlapping classes, fine-grained labels, and supporting multilingual content. It introduces new metrics and applies models like FCN and DeepLab. WiSe [8], the first wild slide segmentation dataset, includes 1,300 annotated slides reflecting real-world conditions like noise and lighting. CSNS [13] aids visually impaired students by segmenting and narrating slide content in reading order.

Due to the high cost of annotating real slides, synthetic alternatives are needed. DreamStruct [20] generates 10,053 synthetic slide-code pairs using UI templates, showing their utility via comparisons with FitVid [14]. While most works focus on pixel-level segmentation, limiting semantic understanding, few explore object-level detection [20]. However, these lack benchmarks for open-world, densely structured lecture slides, instead targeting simpler talking presentations.

Multi-modal Retrieval: The Lecture Presentations Multi-modal Dataset [15] includes 9,000 slide images from 180 hours of lecture videos, annotated with figure bounding boxes and aligned audio transcripts. It introduced *figure-to-text* and *text-to-figure* cross-modal retrieval tasks and PolyViLT, a multi-modal transformer trained with a multi-instance learning loss, surpassing VLMs like CLIP [23] and PCME [29] in retrieval tasks. Similarly, Jobin *et al.* [12] proposed a semantic labels-aware transformer model for lecture slide retrieval, enabling natural language and sketch-based searches. The proposed model trained on the LecSD dataset of 50,000 annotated slides outperforms existing methods and sets a new benchmark in slide retrieval.

2.2 Synthetic Data Generation

Synthetic data generation in computer vision began with statistical methods and evolved through generative modeling to address data scarcity [30], privacy [19], robustness [28], and domain adaptation [25]. In document understanding, it enables the creation of articles [26], layouts [9], handwritten notes [3], and full documents like scientific papers [22] and news reports [27]. Peng *et al.* [20] introduced synthetic user interface (UI) and slide generation pipelines for tasks like image captioning and element recognition, assessing slides with FitVid [14], a

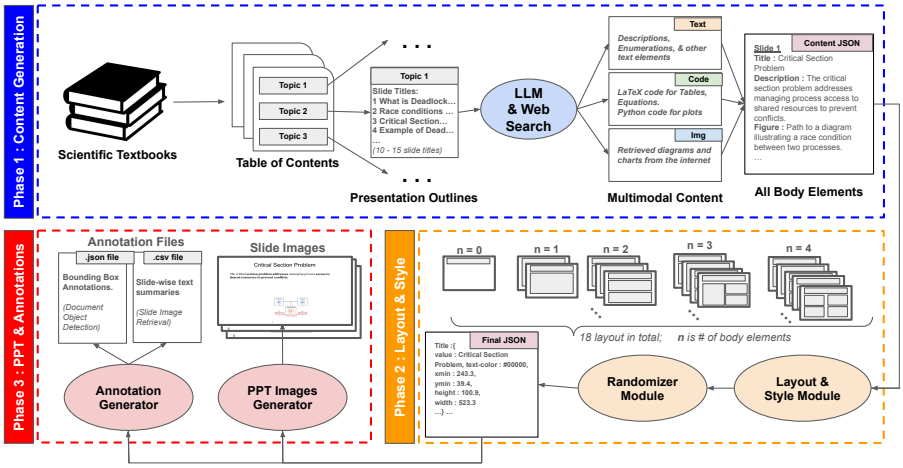


Fig. 1: Overview of our Synthetic Lecture Slide Generation **SynLecSlideGen** pipeline. In Phase I, we generate presentation content using LLMs and Web Image search agents in a multi-step process. In Phase II, we arbitrarily assign layout and style based on each slide's content. Finally, in Phase III, we generate the PPT files, convert them to images, and also generate automatic annotations for multiple downstream tasks from the final JSON file.

dataset of screen-captured video presentations. Likewise, Seng *et al.* [24] developed a pipeline to generate lecture slides from Wikipedia, demonstrating that mixed training with both real and synthetic slides enhances performance on FitVid [14] and SlideVQA [31] for document object detection.

However, studies often overlook fidelity and diversity gaps between synthetic and real slides, as seen in diminishing returns when scaling synthetic data [24,35]. Additionally, the lack of open, multi-task benchmarks limits generalization, as models trained for a single task often fail across others [20].

3 Synthetic Lecture Slide Generation Approach

In order to reduce the reliance on manual annotation required for training lecture slide-related tasks, we propose a large language model (LLM) guided Synthetic Lecture Slide Generation (**SynLecSlideGen**) pipeline consisting of the following three phases. In **Phase I: Content Generation**, we generate semantically rich multi-modal content using LLM and Web image search. In **Phase II: Layout and Style Assignment**, a layout and style discriminator aligns layouts with content types, producing a JSON file that includes all content and layout details to generate a presentation. Finally, in **Phase III: Slide Generation and Annotation**, we create PowerPoint presentation (PPT) files from JSON and convert them to images with automatically generated annotations, elimi-

nating manual effort. Fig. 1 illustrates our three-phase process for high visual quality and coherent lecture slide generation.

3.1 Phase I: Content Generation for Synthetic Lecture Slides

The phase evolves in the following five steps.

Topic Generation: We create a corpus of lecture topics using indexes from openly available textbooks. We select well-known *STEM* textbooks in *Computer Science, Mathematics, Physics, and Economics* such as *Introduction to Algorithms by Thomas Corman, Macroeconomics by Greg Mankiw, etc.* Since large foundational models have been pre-trained on open scientific books, this approach minimizes dependence on local content retrieval as we rely on the model's parametric knowledge [4]. For each textbook, an LLM-based topic generation module produces a list of lecture topics based on the index of contents. For example, *Divide-and-Conquer Approach, Growth Theory in Macroeconomics, etc.* These topics are then used as seeds to generate an entire presentation on each topic in the subsequent steps. We provide a one-shot example using one textbook and its expected topics for each of the four domains before generation. On average, the LLM provides between 10 to 15 topics per textbook, providing seed for as many presentations as possible.

Outline Generation: Given each lecture topic, we prompt the LLM to generate an outline of the presentation, which provides the topic of discussion for each slide in the lecture. The input to this module is the lecture topic (e.g., *Divide-and-Conquer Approach*), and the expected output is a list describing the topic of discussion for each slide (e.g., *Introduction to Divide and Conquer, Motivation behind Divide and Conquer, Pseudo-code for Divide-and-Conquer algorithms, etc*). We utilize three topic-outline pairs from real lectures to elicit in-context learning using few-shot examples. To limit redundancy, we limit the LLM to generate up to 15 slide suggestions per topic. On average, the LLM generates up to 12 to 15 discussion topics per lecture topic, and these are used as *Titles* for each slide in the presentation.

Element Type Generation: This module determines what elements are suitable for each slide based on the title. We use Chain-of-thought [34] prompting technique to instruct the LLM first to determine the type of lecture slide based on the slide title and then suggest the type of elements based on the slide type. Following previous work [16], we consider six types of lecture slides – *Introduction, Definition, Example, Comparison, Conclusion, and References.* Intuitively, an *Introduction* slide may contain text and/or enumeration with an optional figure. A *Comparison* slide should ideally have a table for comparison, etc. We also provide examples of three real lecture slides of {title, type, elements} tuples for better suggestions. Hence, the input to this module is the slide title, and the output is a list of tuples where each tuple is {element, caption}. *For instance, an example slide on Divide-and-Conquer approach will have {"code", "a pseudo*

code example of Divide and Conquer"} as one of the suggestions. As is generally observed in real slides, we ask the LLM to generate up to three body element suggestions per slide. The output is divided into four prompt types: *text* generation, *figure* retrieval, *structure* generation, and *plot* generation. These prompts are provided as separate chains to the LLM to maximize the output tokens generated.

Textual Element Generation: The text generation prompt includes *Description and Enumeration* elements where the LLM outputs the text content of each, based on the provided caption by the previous step.

Structural Elements Generation and Diagram Retrieval: Structural elements like *Tables, Equations* and plot elements like *Charts* are generated using *LaTeX* and *Python* code, respectively. The code is temporarily stored and later compiled into image-based graphics. Other visual elements like *Diagrams* are challenging to generate using LLMs. Hence, they are retrieved from the internet by using the *Bing Search API*[4] based on the element caption as a search query. We fetch the top two related diagrams, with one selected randomly for inserting in the slide.

The output from steps 4 and 5 is compiled into a JSON file containing text content and image file hyperlinks for the presentation. We use a combination of GPT-3.5-Turbo and GPT-4 base APIs for slide content generation[5]. The top part of Fig. 1 illustrates Phase I of the pipeline.

3.2 Phase II: Layout and Style Assignment

This phase assigns slide-level layout and styling properties for each slide in the presentation. It consists of a Python module that assigns one of 18 predefined slide layouts based on the size and count of body elements. We utilize 9 of 11 layouts available in the python-pptx library[6] such as *two-column*, *one-row-two-column*, etc. and generate one counterpart for each where *Title* element is absent. In addition, we arbitrarily assign style attributes such as background color, font styling for text, design templates, shadows, and borders. Meta elements, e.g., *slide numbers, footers, natural images, graphics, and logos* are randomly inserted to mimic real slides. Styling for certain elements *(e.g., font styling for title)* is kept consistent throughout the presentation while allowing slide-wise variations for other elements *(e.g., description text color)*. To introduce variability, we implement two perturbation functions by (i) applying Gaussian noise to slightly shift element positions within layout boundaries and (ii) modifying elements by their styling. The resulting JSON file now contains all the content, style, and layout information required to generate the presentation. We also use this file to

[4] https://www.microsoft.com/en-us/bing/apis/bing-web-search-api

[5] Complete details of the LLMs used at each stage, along with the specific prompts, are provided in Appendix A of the supplementary material.

[6] https://python-pptx.readthedocs.io/en/latest/dev/analysis/sld-layout.html

automatically extract annotations described in the next stage. The middle part of Fig. 1 illustrates the steps of Phase II.

3.3 Phase III: Lecture Slide Generation and Obtaining Automatic Annotation

This phase constructs the PPT file and generates annotations for downstream tasks using the JSON file obtained from the previous phase. We utilize *python-pptx library*[7] that enables generating *".PPT"* files from structured data like XML or JSON. The PPT file is then converted into a sequence of slide images. We use data augmentation techniques, like Gaussian blurs, pixelation, and resizing, to increase the variability of the obtained images. We utilize the data from the JSON file described below to obtain annotations.

Slide Element Detection: We utilize the bounding-box coordinates for each slide element (e.g., title, table, figure, etc.) from the JSON file and their labels to construct a ground truth file compatible with the COCO [18] object detection annotation format for the slide element detection tasks.

Slide Image Retrieval: We extract content details (e.g., titles, descriptions, enumerations), layout attributes (e.g., element sizes), spatial positions (e.g., top of the slide), and meta elements (e.g., footers, URLs) from the JSON file to generate slide-level summaries for the slide image retrieval task. The output is a CSV file containing image IDs and their corresponding summaries.

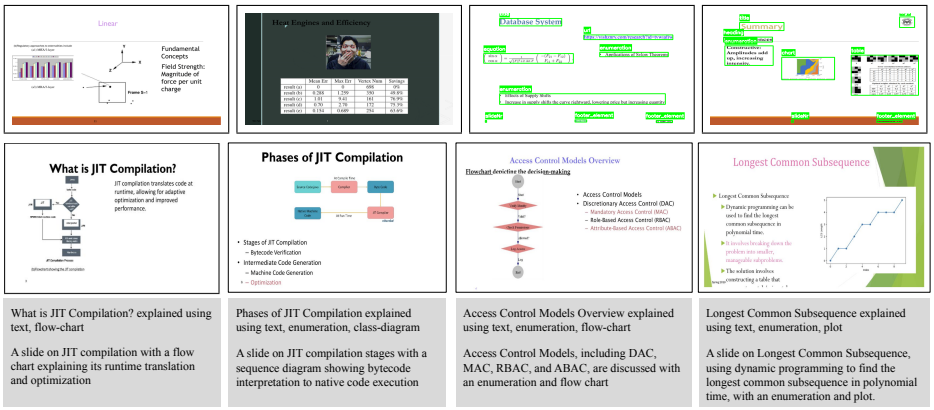
The final dataset consists of synthetically generated slide images and annotation files. The leftmost part of the bottom row of Fig. 1 illustrates the final phase of the pipeline.

4 Dataset

4.1 SynSlide

Our synthetic slide dataset, **SynSlide** is created using our pipeline explained in Section 3. It comprises two subsets: **SynDet** for document layout analysis and **SynRet** for text-based slide retrieval. **SynDet** includes 2,200 high quality slide images with automatic bounding box annotations for 16 element categories namely *(Title, Description, Enumeration, SlideNr, Equation, Table, Logo, Heading, Diagram, Chart, Footer-Element, Code, Figure-Caption, Table-Caption, URL, Natural-Image)*. **SynRet** contains 2,200 topic coherent slide images with two types of slide summaries for each slide. Both of the synthetic sets have been constructed using the same generation pipeline with some task-specific processing steps as defined in Fig. 1. A few examples from **SynDet** and **SynRet** are presented in Fig. 2.

[7] https://pypi.org/project/python-pptx/



What is JIT Compilation? explained using text, flow-chart

A slide on JIT compilation with a flow chart explaining its runtime translation and optimization

Phases of JIT Compilation explained using text, enumeration, class-diagram

A slide on JIT compilation stages with a sequence diagram showing bytecode interpretation to native code execution

Access Control Models Overview explained using text, enumeration, flow-chart

Access Control Models, including DAC, MAC, RBAC, and ABAC, are discussed with an enumeration and flow chart

Longest Common Subsequence explained using text, enumeration, plot

A slide on Longest Common Subsequence, using dynamic programming to find the longest common subsequence in polynomial time, with an enumeration and plot.

Fig. 2: Displays sample slides from our generated **SynSlide** dataset. The first row presents examples from **SynDet**, with and without automatic bounding box annotations for elements. The second row showcases coherent slides from **SynRet**. The third row shows two types of summaries of corresponding slides.

Most slide image retrieval research [12,20] focuses on retrieving images based on slide transcripts. LecSD [12] enhances retrieval capabilities by supporting both text and sketch-based queries related to slide content and visuals. We use structured content files generated from synthetic slide creation to produce two types of summaries. The first type follows the *LecSD-style* query format, where slide titles or enumerations are explained using key elements. This format, created via OCR, is sensitive to variations in query phrasing. To overcome this limitation, we introduce an OCR-free semantic summaries generated using a large language model (LLM) that incorporates slide content, type of slide elements, and metadata *like name of instructor in footer, etc.*, thereby making slide retrieval more robust.

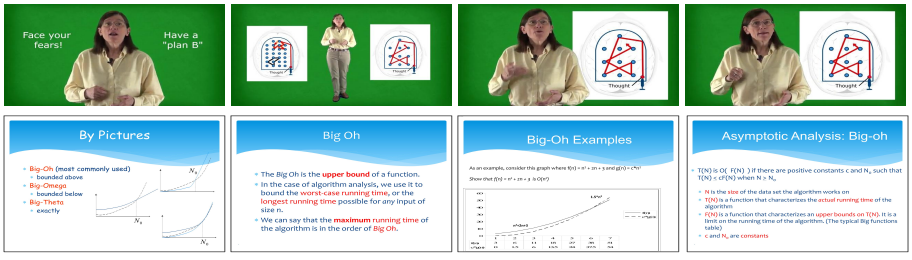


Fig. 3: Comparison of four consecutive slides from lecture slide datasets. The top row displays slide images from the existing FitVid [14] dataset, while the bottom row presents slide images from our **RealSlide**.

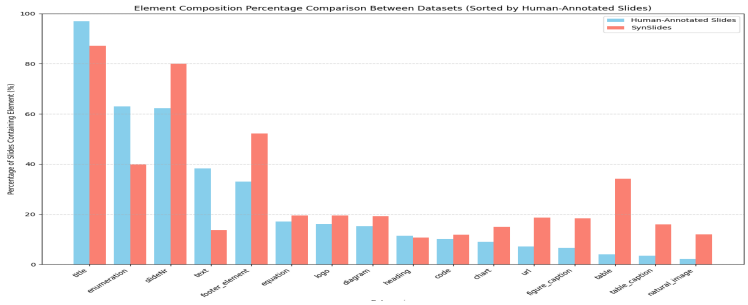


Fig. 4: Occurrence of each element in the **SynDet** (Red) and **RealSlide** (Blue) datasets. **SynDet** contains twice as many elements for the five scarcest classes in **RealSlide**: URL, Figure Caption, Table, Table Caption, and Natural Image.

Table 1: Comparison between **RealSlide** and **SynSlide**. Abbreviations: Des. = Description, Ele. = Element, Nat. = Natural, Img. = Image, Cap. = Caption.

Element	**RealS.**	**SynS.**
#Slide	1050	4400
#Des. Ele.	1109	3518
#Figures	484	3862
#Meta Ele.	1010	6640
Avg. Ele. per Slide	4.77	5.82

(a) Key statistics between **RealSlide** and **SynSlide** (SynRet + SynDet)

Element	**RealS.**	**SynS.**
Nat. Img.	33	700
Table Cap.	47	602
URL	61	892
Table	69	2012
Heading	106	834

(b) Comparison of top five scarce elements in **RealSlide** and **SynSlide**.

4.2 RealSlide

Existing benchmarks for slide object detection either provide pixel-level annotations [7,8], originate from talking presentations [14], or lack detailed semantic annotations — for instance, grouping multiple bullet points into a single enumeration block or failing to distinguish between distinct figures [7,8,31]. Moreover, current real datasets only provide annotations for a single task which limits evaluating document models on multiple tasks. Because of this, a new real benchmark for evaluation is needed. For this purpose, we curate a new benchmark, **RealSlide**, with 1050 human-annotated pre-made lecture slides for detection and retrieval tasks. These slides are curated using real lecture presentations[8] delivered for university-level courses in the fields of Computer Science (50%), Economics (20%), Physics (15%), and Mathematics (15%). Fig. 3 compares samples from an existing lecture slide dataset [14] and our **RealSlide**. **RealSlide** is used as a validation set to evaluate the performance of **SynSlide** on two down-

[8] Publicly available lecture slides distributed under Creative Commons (CC) license. Links to all presentations are listed in supplementary material.

stream tasks. The dataset is divided into a training set of 300 slides (30%) for fine-tuning and a validation set of 750 slides (70%) for evaluation. To prevent data leakage across different sets, we systematically partition the data to ensure that slides from the same presentation are exclusive to training or validation sets. Fig. 4 illustrates the percentage of element occurrences in **RealSlide** and **SynSlide**, while Table 1 compares element-wise statistics between **RealSlide** and **SynSlide**

Table 2: Comparison of lecture datasets with various features and availability. (M) indicates manually annotated data, while (A) indicates automatically annotated data. * after pruning classes like handwritten text, maps, etc. † only images are available.

	Features		**Annotations**		**Size**		**Avail.**
	Multi-page	Semantic	Layout	Summary	#BBox Class	#Slide	
Real Lecture Slide Datasets							
SPaSe [7]			✓(M)		18*	2000	✓
WiSe [8]			✓(M)		18*	1300	✓
FitVid [14]	✓		✓(A)		12	5500	✓
LecSD [12]				✓(M/A)	-	54000	✓
MLP [15]	✓		✓(M/A)*	✓(M/A)	6	9031	✓
RealSlide (Ours)	✓	✓	✓(M)	✓(M)	16	1050	✓
Synthetic Lecture Slide Datasets							
DreamStruct [20]			✓(A)	✓(A)	12	10053	✓†
SlideCraft [24]	✓	✓	✓(A)		12	25000	
SynSlide (Ours)	✓	✓	✓(A)	✓(A)	16	4400	✓

4.3 Comparison with Existing Lecture Slide Datasets

We compare the generated **SynSlides** and evaluation set **RealSlides** with existing lecture slide datasets, focusing on their applicability to real-world tasks such as slide narration and real-time slide generation. Table 2 compares current real and synthetic benchmarks against our proposed datasets. To assess the quality of our generated **SynSlide** in comparison to real slides and other synthetically produced slides, such as Dreamstruct, we utilize the Fréchet Inception Distance (FID) [10]. The FID score is a technique to quantify the distance between two distributions. A lower score indicates a higher resemblance and vice versa. Table 3 compares the FID scores of two synthetic datasets, DreamStruct[20] and SynSlide against RealSlide. We also compare two equally-sized, presentation-wise split sets of RealSlide as control. The result indicates that SynSlide is closer in distribution to the real benchmark than recently proposed DreamStruct dataset.

Table 3: FID scores for several datasets.

Dataset 1	Dataset 2	FID Score ↓
RealSlide	RealSlide	18.4
SynSlide	RealSlide	42.5
Dreamstruct	RealSlide	56.1

5 Experiments

5.1 Slide Element Detection

Baselines and Implementation Details: We employ three popular models for slide element detection: LayoutLMV3 [11], YOLOV9 [33], and DETR [5]. For each of the models, we follow two different fine-tuning strategies — (i) **Single Stage (SS):** pre-trained model is fine-tuned with the training set of **RealSlide** dataset and (ii) **Two Stage (TS):** pre-trained model is fine-tuned with our synthetic **SynDet** and again fine-tuned with the training set of **RealSlide** dataset to adapt real-world slide layouts. For the LayoutLMV3 and YOLOV9 models, we initialize the weights using the pre-trained models from PubLayNet [36] and DocLayNet [21], respectively. For the DETR model, we utilize the publicly available COCO pretrained checkpoint . We set the batch size to 16 for all models and follow the standard settings specific to each model keeping all parameters trainable. Additionally, we resize all images to 360×640 pixels and do not apply any image augmentations.

Table 4: Impact of the IoU threshold on slide element detection performance (mAP) for three baseline models under two fine-tuning strategies: (i) **Single Stage** (**SS**) and (ii) **Two Stage** (**TS**) on the test set of the **RealSlide** dataset.

IoU	**YOLOV9**		**LayoutLMV3**		**DETR**	
	SS	**TS**	**SS**	**TS**	**SS**	**TS**
mAP@[0.50]	50.3	53.4	38.9	49.0	36.6	40.9
mAP@[0.55]	49.2	51.8	37.6	47.9	33.4	39.1
mAP@[0.60]	48.0	50.1	36.2	46.5	30.9	37.7
mAP@[0.65]	46.2	48.4	34.0	42.0	26.1	35.6
mAP@[0.70]	43.6	46.0	30.8	39.2	22.4	30.4
mAP@[0.75]	39.9	42.5	27.2	33.9	18.5	26.9
mAP@[0.80]	34.6	36.3	23.9	28.4	14.6	20.7
mAP@[0.85]	26.1	29.4	20.3	23.6	10.8	17.0
mAP@[0.90]	18.5	19.7	14.4	17.2	08.5	11.8
mAP@[0.95]	11.3	10.5	08.8	10.1	04.4	08.5
mAP@[0.5-0.95]	36.8	38.8	27.3	33.9	21.2	27.0

Effect of IoU: To assess the impact of the IoU threshold on slide element detection performance, we compute mAP across multiple IoU thresholds from

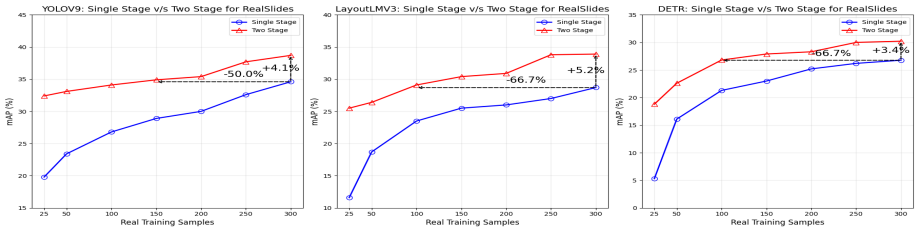


Fig. 5: Effect of real slide images (RealSlide) on the performance (mAP@[0.5–0.95]) of three slide element detection models under two training strategies: (i) **Single Stage** and (ii) **Two Stage**.

0.5 to 0.95. Table 4 presents the performance of slide detection models under these thresholds.

Effect of Real Slide Images: In real-world scenarios, obtaining a large number of slide images with region bounding box annotations is challenging and expensive. At the same time, annotation using synthetic data is cheap but often lacks enough variance as seen in real world data. Therefore, it is essential to determine the amount of real annotated slide images required to improve the performance of slide element detection models. To explore this, we conduct experiments under two fine-tuning strategies: (i) **Single Stage** — the pre-trained model is directly fine-tuned with real annotated slide images and (ii) **Two Stage** — the pre-trained model is first fine-tuned with synthetic slide images, followed by fine-tuning with real annotated slide images. We randomly select multiple subsets of 25, 50, 100, 150, 200, 250, and 300 images from the real training set and fine-tune the three detection models under both strategies. Fig. 5 and Fig. 6 presents the mAP results for different models on increasing subsets of training with **RealSlide** and **FitVid** data respectively. From the results, we observe that increasing the number of real training slide images consistently and naturally enhances model performance. Additionally, the **Two Stage** strategy provides a greater performance boost compared to **Single Stage**, demonstrating that fine-tuning with synthetic slide images before real slide images leads to a marginal yet meaningful improvement.

Results Analysis: Table 5 presents the element-wise mean Average Precision (mAP) scores across models for both single-stage and two-stage fine-tuning under two experimental conditions: (i) a low-resource setting with only 50 real images and (ii) a higher-resource setting with 300 real images. Notably, the two-stage fine-tuning approach leads to a substantial performance boost in low-frequency classes such as **Natural Image**, **Table Caption**, and **Code**. This improvement underscores the efficacy of synthetic data in enhancing model generalization under data-scarce conditions, as further illustrated in Fig. 4.

The confusion matrix in Fig. 6 (left) reveals a high frequency of misclassifications among semantically similar classes, particularly between *diagram* and

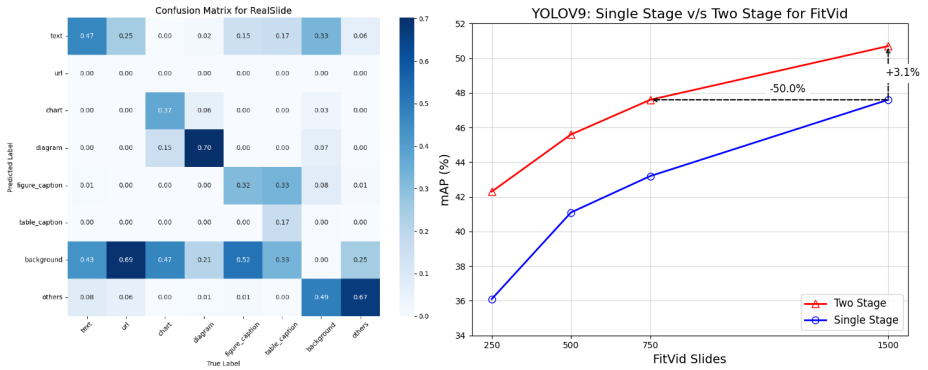


Fig. 6: Left: Confusion Matrix for semantically similar classes. Right: Effect of real slide images (FitVid) on the performance (mAP@[0.5–0.95]) of the best model (YOLOV9) for two training strategies: (i) **Single Stage** and (ii) **Two Stage**.

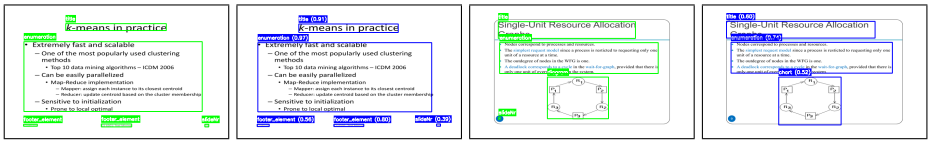


Fig. 7: Illustration of a selection of visual results from YOLOv9 (Two-Stage), where green denotes ground truth bounding boxes and blue indicates predicted bounding boxes.

chart. Furthermore, several text-based categories — including *URL*, *heading*, *footer element*, *figure caption*, *table caption*, and *code* — are frequently misclassified as the generic *"Text"* class. This phenomenon can be attributed to both intra-class variability (e.g., differing styles of diagrams and charts) and inter-class feature overlap (e.g., structural and contextual similarities between URLs, headings, and table captions).

Fig. 7 provides qualitative examples of slide element detection, further illustrating the strengths and limitations of the proposed approach. These findings highlight the importance of leveraging synthetic data and multi-stage fine-tuning to mitigate class imbalance and enhance recognition performance in real-world document analysis tasks.

5.2 Text-based Slide Retrieval

Baselines and Implementation Details: We consider CLIP [23] as our baseline for retrieval tasks. We define the task as retrieving lecture slide images based on slide summary captions. We implement ViT-B/32 CLIP model and use 128 batch size keeping all other parameters standard. While finetuning, we set all

Table 5: Element-wise mAP @ IoU [0.50:0.95] for three slide element detection models under two fine-tune strategies: (i) **Single Stage** (**SS**) and (ii) **Two Stage** (**TS**) on the test set (750 images) of the **RealSlide** dataset.

Element	**Fine-tuning using 50 Real Images**						**Fine-tuning using 300 Real Images**					
	YOLOV9		**LayoutLMV3**		**DETR**		**YOLOV9**		**LayoutLMV3**		**DETR**	
	SS	**TS**	**SS**	**TS**	**SS**	**TS**	**SS**	**TS**	**SS**	**TS**	**SS**	**TS**
Title	69.1	75.3	66.2	71.9	57.8	67.2	70.9	75.4	71.8	77.0	59.5	69.7
Text	06.5	17.4	11.3	13.2	07.4	13.8	15.6	21.5	13.8	18.1	09.1	15.3
Enumeration	57.5	66.8	60.7	67.8	67.1	72.4	70.9	76.7	72.0	79.9	68.6	74.9
URL	00.0	03.4	00.0	00.8	00.8	01.5	03.4	01.3	02.4	02.1	01.8	02.7
Equation	09.2	28.3	00.9	09.8	16.6	20.2	23.0	27.4	16.5	24.2	18.3	22.1
Table	57.8	60.1	35.7	43.3	48.5	50.6	82.7	56.2	59.0	63.1	56.5	51.3
Diagram	25.8	46.3	26.6	39.9	33.7	41.0	53.8	58.6	46.0	50.4	38.8	44.7
Chart	12.8	33.4	08.5	17.4	08.9	14.4	31.7	32.1	14.9	23.1	11.7	18.9
Heading	06.1	09.2	03.8	16.1	10.6	18.8	18.6	22.8	24.6	35.2	13.1	20.3
Slide Number	25.0	27.3	33.4	29.3	20.8	24.1	27.7	25.9	28.2	26.7	22.5	25.6
Footer Element	48.7	42.2	47.2	48.0	36.6	42.0	51.5	47.7	43.0	48.9	40.0	45.1
Figure caption	02.7	05.7	00.3	10.9	01.8	06.7	15.8	14.2	07.6	09.8	04.4	08.1
Table caption	00.0	11.8	00.0	02.0	01.3	06.9	19.2	21.6	00.0	02.2	02.1	08.7
Logo	48.3	46.1	03.7	28.1	18.8	26.2	67.9	69.4	26.0	42.9	22.6	34.7
Code	02.1	34.6	00.0	05.0	08.0	14.5	23.8	42.5	10.5	18.6	12.1	17.8
Natural Image	00.7	20.9	00.0	12.0	00.4	11.7	11.8	27.9	10.4	18.3	09.3	14.6
Macro avg	23.3	33.0	18.6	26.1	21.2	27.0	36.8	38.8	27.9	33.8	26.8	30.2

parameters to be trainable. Table 6 demonstrates experiments comparing publicly available slide image-caption pairs along with SynRet and RealSlide train data evaluated on LecSD and RealSlide test sets.

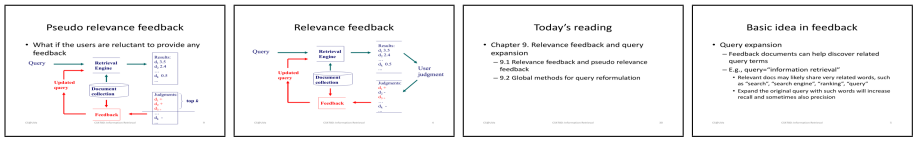
Results Analysis: We present text-to-lecture slide retrieval results (summarized in Table 6) on two benchmarks: LecSD-Test, which comprises 10,000 lecture slides, and our newly annotated RealSlide dataset containing 750 images. Our results show that the synthetic dataset is especially useful when in-domain real slide annotations are unavailable. For instance, it achieves an R@1 of 43 on the RealSlide set, outperforming fine-tuning with out-of-domain data like LecSD-Train. Additionally, our synthetic data provides a slight improvement in R@1 compared to other synthetic datasets, such as DreamStruct [20], highlighting the effectiveness of our approach in generating useful training data for lecture slide retrieval. We also train the CLIP model using the same two-stage fine-tuning strategy defined in the Slide Element Detection task; however, minimal improvement is observed in the performance over SynRet alone. Fig. 8 shows a qualitative result from the RealSlide (750) test set, where the fine-tuned model using SynRet correctly retrieves the relevant slide, with the top-3 results also closely matching the query.[9].

[9] More generated sample slide images and qualitative results are given in supplementary material.

Table 6: Text-based Lecture Slide Retrieval using CLIP model. We show Recall@1 and Recall@10.

Dataset			R@1	R@10
Finetuning dataset (# Samples)	**Test Dataset (# Samples)**	**In-domain**		
None (zero-shot)	LecSD-Test (10,000)	NA	16	44
LecSD-Train (31,475)		Yes	45	78
DreamStruct (3,183)		No	26	59
SynRet (2,200)		No	26	60
RealSlide (300)		No	20	49
None (zero-shot)	RealSlide (750)	NA	33	63
LecSD-Train (31,475)		No	31	57
DreamStruct (3,183)		No	42	67
SynRet (2,200)		No	43	69
RealSlide (300)		No	40	69
SynRet(2200) + RealSlide (300)		No	43	70

Query: A slide on Pseudo relevance feedback with a diagram and enumeration



(a) 1st (b) 2nd (c) 3rd (d) 4th

Fig. 8: Visual example of result from the CLIP model fine-tuned using SynRet data

6 Conclusion and Discussion

This paper presents an open-source, LLM-driven pipeline to generate realistic, coherent, copyright-free lecture slides with automatic annotations, resulting in the **SynSlide** dataset for slide understanding. We also introduce **RealSlide**, an evaluation set of 1,050 manually annotated real slides for benchmarking element detection and text-based retrieval. Experiments show that few-shot transfer learning on real slides using models pre-trained on synthetic slides outperforms training on real data alone, emphasizing the value of synthetic slides in low-annotation scenarios.

We find that vision models struggle with fine-grained document tasks, exhibiting high class confusion (Fig. 6) and limited gains from more training data (Table 5). While our synthetic slides help improve performance across multiple tasks with no manual annotation, increasing synthetic data does not lead to proportional gains due to limited data variability. Future work may explore diffusion-based layout generation, context integration, and better generalization. This highlights the rising importance of synthetic data for document understanding.

Acknowledgments

This work is supported by the MeitY Government of India, through the NLTM Bhashini (https://bhashini.gov.in/) project.

References

1. Appalaraju, S., Jasani, B., Kota, B.U., Xie, Y., Manmatha, R.: Docformer: End-to-end transformer for document understanding. In: ICCV. pp. 993–1003 (2021) 2
2. Araujo, A., Chaves, J., Lakshman, H., Angst, R., Girod, B.: Large-scale query-by-image video retrieval using bloom filters. arXiv preprint arXiv:1604.07939 (2016) 3
3. Blanes, A.R.: Synthetic handwritten text generation. Univ. Autònoma de Barcelona (2018) 3
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. NeurIPS **33**, 1877–1901 (2020) 5
5. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV. p. 213–229 (2020) 11
6. Da, C., Luo, C., Zheng, Q., Yao, C.: Vision grid transformer for document layout analysis. In: ICCV. pp. 19462–19472 (2023) 2
7. Haurilet, M., Al-Halah, Z., Stiefelhagen, R.: Spase-multi-label page segmentation for presentation slides. In: WACV. pp. 726–734 (2019) 2, 3, 9, 10
8. Haurilet, M., Roitberg, A., Martinez, M., Stiefelhagen, R.: Wise—slide segmentation in the wild. In: ICDAR. pp. 343–348 (2019) 2, 3, 9, 10
9. He, L., Lu, Y., Corring, J., Florencio, D., Zhang, C.: Diffusion-based document layout generation. In: ICDAR. pp. 361–378 (2023) 3
10. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: NeurIPS. p. 6629–6640 (2017) 10
11. Huang, Y., Lv, T., Cui, L., Lu, Y., Wei, F.: Layoutlmv3: Pre-training for document ai with unified text and image masking. In: ACM MM. pp. 4083–4091 (2022) 2, 11
12. Jobin, K., Mishra, A., Jawahar, C.: Semantic labels-aware transformer model for searching over a large collection of lecture-slides. In: WACV. pp. 6016–6025 (2024) 2, 3, 8, 10
13. Jobin, K., Mondal, A., Jawahar, C.: Classroom slide narration system. In: CVIP. pp. 135–146 (2021) 3
14. Kim, J., Choi, Y., Kahng, M., Kim, J.: Fitvid: Responsive and flexible video content adaptation. In: ACM CHI. pp. 1–16 (2022) 3, 4, 8, 9, 10
15. Lee, D.W., Ahuja, C., Liang, P.P., Natu, S., Morency, L.P.: Multimodal lecture presentations dataset: Understanding multimodality in educational slides. arXiv (2022) 2, 3, 10
16. Li, I., Fabbri, A.R., Tung, R.R., Radev, D.R.: What should i learn first: Introducing lecturebank for nlp education and prerequisite chain learning. In: AAAI. vol. 33, pp. 6674–6681 (2019) 5
17. Li, M., Xu, Y., Cui, L., Huang, S., Wei, F., Li, Z., Zhou, M.: Docbank: A benchmark dataset for document layout analysis. arXiv preprint arXiv:2006.01038 (2020) 2

18. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. pp. 740–755 (2014) 7
19. Osuala, R.: Enhancing the utility of privacy-preserving cancer classification using synthetic data. In: MICCAI Workshop. p. 54 (2024) 3
20. Peng, Y.H., Huq, F., Jiang, Y., Wu, J., Li, X.Y., Bigham, J.P., Pavel, A.: Dreamstruct: Understanding slides and user interfaces via synthetic data generation. In: ECCV. pp. 466–485 (2024) 3, 4, 8, 10, 14
21. Pfitzmann, B., Auer, C., Dolfi, M., Nassar, A.S., Staar, P.: DocLayNet: A large human-annotated dataset for document-layout segmentation. In: ACM SIGKDD (2022) 11
22. Pisaneschi, L., Gemelli, A., Marinai, S.: Automatic generation of scientific papers for data augmentation in document layout analysis. Pattern Recognition Letters **167**, 38–44 (2023) 3
23. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021) 3, 13
24. Seng, T., Carlier, A., Forgione, T., Charvillat, V., Ooi, W.T.: Slidecraft: Synthetic slides generation for robust slide analysis. In: ICDAR. pp. 79–96 (2024) 4, 10
25. Shakeri, S., Santos, C.N.d., Zhu, H., Ng, P., Nan, F., Wang, Z., Nallapati, R., Xiang, B.: End-to-end synthetic data generation for domain adaptation of question answering systems. arXiv (2020) 3
26. Shao, Y., Jiang, Y., Kanell, T.A., Xu, P., Khattab, O., Lam, M.S.: Assisting in writing wikipedia-like articles from scratch with large language models. arXiv preprint arXiv:2402.14207 (2024) 3
27. Shu, K., Li, Y., Ding, K., Liu, H.: Fact-enhanced synthetic news generation. In: AAAI. vol. 35, pp. 13825–13833 (2021) 3
28. Singh, K., Navaratnam, T., Holmer, J., Schaub-Meyer, S., Roth, S.: Is synthetic data all we need? benchmarking the robustness of models trained with synthetic images. In: CVPR. pp. 2505–2515 (2024) 3
29. Song, Y., Soleymani, M.: Polysemous visual-semantic embedding for cross-modal retrieval. In: CVPR. pp. 1979–1988 (2019) 3
30. Sufi, F.: Addressing data scarcity in the medical domain: A gpt-based approach for synthetic data generation and feature extraction. Information **15**(5), 264 (2024) 3
31. Tanaka, R., Nishida, K., Nishida, K., Hasegawa, T., Saito, I., Saito, K.: Slidevqa: A dataset for document visual question answering on multiple images. In: AAAI. pp. 13636–13645 (2023) 2, 4, 9
32. Tang, Z., Yang, Z., Wang, G., Fang, Y., Liu, Y., Zhu, C., Zeng, M., Zhang, C., Bansal, M.: Unifying vision, text, and layout for universal document processing. In: CVPR. pp. 19254–19264 (2023) 2
33. Wang, C.Y., Yeh, I.H., Mark Liao, H.Y.: YOLOV9: Learning what you want to learn using programmable gradient information. In: ECCV (2024) 11
34. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. NeurIPS **35**, 24824–24837 (2022) 5
35. Yu, Z., Zhu, C., Culatana, S., Krishnamoorthi, R., Xiao, F., Lee, Y.J.: Diversify, don't fine-tune: Scaling up visual recognition training with synthetic images. arXiv preprint arXiv:2312.02253 (2023) 4
36. Zhong, X., Tang, J., Yepes, A.J.: PubLayNet: largest dataset ever for document layout analysis. In: ICDAR (2019) 2, 11