The University of Hong Kong

Faculty of Engineering

Department of Computer Science

2024 - 2025

COMP 4801 Final Year Project

FYP Final Report (Group)

An Enhanced Note Assistance Application with Handwriting Recognition Leveraging LLM

Gu Zhuangcheng 3035827110

Deng Jiaqi 3035832490

Xie Changhe 3035770575

Zhou Zihan 3035772640

Supervisor: Prof. Luo Ping

Date of submission: Apr 21st, 2025

Abstract

The increasing reliance on digital note-taking in academic and professional scenarios highlights the need for efficient AI-driven solutions capable of accurately processing diverse types of documents. In this report, we present an AI-powered note-taking application that leverages Vision-Language Models (VLMs) to extract and process text, images, and multimedia in both scientific documents and handwritten drafts. The system was built using Siglip ncoder and Qwen2 decoder as the baseline model, which was fine-tuned on a diverse and extensive dataset that combined real-world public datasets and strategically generated synthetic data. Synthetic data generation tools, such as LaTeX and Synthdog, were utilized to create examples of text, mathematical formulas, tables, and charts to enhance model versatility. Our fine-tuning results have shown significant improvements in multiple tasks, including visual grounding, text retrieval, and understanding multimedia content such as figures and formulas. Designed as a comprehensive note-taking solution, this application aims to streamline academic and professional workflows by automating the extraction, processing, and synthesis of complex content. Future enhancements will focus on expanding its functionalities and optimizing performance for broader usability across different devices and contexts.

Acknowledgment

We would like to express our sincere gratitude to Prof. Luo Ping and Prof. Liang Yingyu for their invaluable instruction and guidance during our preliminary research and for their insightful ideas that greatly contributed to the planning of our project. We also extend our thanks to the creators of various public datasets, such as DVQA, OCR-VQA, and others, whose work provided a strong foundation for our dataset. Finally, we are grateful to Llava for providing a public repository that served as the code base for model fine-tuning, which was instrumental in the development of our system.

Table of Contents

Abstract i

Acknowledgment ii

List of Figures vi

List of Tables vii

Abbreviations viii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Objectives and Deliverables . . . 3
- 1.3 Contributions . . . 4

2 Project Background 6
- 2.1 Background . . . 6
- 2.2 Document Parsing Approaches . . . 8
  - 2.2.1 Traditional OCR . . . 8
  - 2.2.2 Deep Learning Based Recognition . . . 10
  - 2.2.3 Recent Advancement with VLM . . . 10
- 2.3 Document Question Answering . . . 11
- 2.4 Related Datasets and Metrics . . . 11
  - 2.4.1 OCR Datasets . . . 11
  - 2.4.2 DocQA Datasets . . . 12
- 2.5 Modern Application Development . . . 13

3 Methodology 14
- 3.1 Overall Model Design . . . 15
- 3.2 Document Parsing and VQA Model . . . 16
  - 3.2.1 Model Architecture . . . 16
  - 3.2.2 Visual Representations . . . 17
  - 3.2.3 Model Fine-tuning . . . 18
- 3.3 Data . . . 19

3.3.1 Dataset Overview . . . 19
3.3.2 Data Collection . . . 21
3.3.3 Data Synthesis . . . 21
3.3.4 Dataset Preprocessing . . . 21
3.3.5 Data Format and Prompt Design . . . 22
3.3.6 Image Augmentation . . . 23
3.4 Application Development . . . 24
3.4.1 System Design . . . 25
3.4.2 Backend Design . . . 25
3.4.3 Model Deployment . . . 26
3.4.4 Frontend & UI Design . . . 26

4 Experiments and Results 28
4.1 Dataset and Evaluation Metrics . . . 28
4.1.1 Dataset . . . 28
4.1.2 Metrics . . . 28
4.2 Model Implementation Details . . . 29
4.3 Comparison with SOTA Methods . . . 30
4.3.1 Document Parsing Model . . . 30
4.4 VQA Model . . . 31
4.5 Ablation Study . . . 32
4.5.1 Model Architecture . . . 32
4.5.2 Removal of Spatial Unpad . . . 33
4.6 Qualitative Results . . . 33
4.7 Backend Implementation Details . . . 35
4.7.1 Backend Architecture . . . 35
4.7.2 Vision Language Model Integration . . . 37
4.7.3 Data Model Design . . . 39
4.8 Desktop Frontend Development . . . 42
4.8.1 Canvas-Based Note-Taking . . . 42
4.8.2 Hierarchical File Management . . . 42
4.8.3 Interactive Model Integration . . . 43
4.8.4 Responsive and Customizable Interface . . . 43
4.9 Mobile Application Frontend Development . . . 44

4.10 Case Studies 44
4.10.1 Case 1: Parsing a Handwritten Math Note 44
4.10.2 Case 2: Understanding a Chart in a Scientific PDF 45

5 Ethical Considerations and Privacy **46**
5.1 User Data Privacy 46
5.2 Bias in Model Predictions 46
5.3 Responsible Use and Transparency 47
5.4 Predictive Privacy 47

6 Conclusion **48**

References

Appendix

List of Figures

1 Comparison of Note-Taking Applications: AI Capabilities and Handwriting Support[1] . . . 2
2 Handwritten notes and reference documents are digitized through OCR and caption extraction, then merged to power an AI assistant that answers user queries . . . 3
3 The Global AI Note Taking Market Apporximation from 2024 - 2033 . . . 6
4 Taxonomy of Document Parsing and Question Answering Approaches . . . 9
5 Methodology Overview. . . . 14
6 The designed architecture of model workflow . . . 15
7 Image Preprocessing and Visual Representation. . . . 17
8 Two-stage fine-tuning workflow. flame and snowflake represent the activation and freezing of the corresponding model components. The first stage focuses on training the MLP projector, while the second stage fine-tunes the entire model end-to-end. . . . 18
9 Dataset Overview of OmniNote-1M . . . 19
10 System Architecture of Our Application. . . . 24
11 User-interface prototypes of the note-taking application on desktop (top) and mobile (bottom) platforms. . . . 27
12 Visualization of Document Parsing Results. . . . 34
13 More Grounding Results. . . . 34
14 JSON representation of a Canvas object, illustrating the serialization of properties and elements into a structured format. . . . 40
15 Design and functionality of the note management system in desktop application . . . 43

List of Tables

1 Representative Datasets for Document Parsing, Handwriting OCR, and Visual Question Answering 12
2 Dataset statistics of *OmniNote-1M*. 20
3 Dataset Format Design for Text, Table, and Equation Tasks. 22
4 Dataset Prompt Design for OCR, VQA Tasks with Grounding Labels. . . . 22
5 Compare to SOTA Methods on our Test Sets. **Bold** and Underlined Scores Refer to Best or Second Best Model. 30
6 Results using Qwen2VL as base model 31
7 Ablation Study of Different Encoder and Decoder Combinations. 32
8 Ablation Study of Replacing Unpad with Resize. 33

Abbreviations

Abbreviation	Definition
OCR	Optical Character Recognition
VDU	Visual Document Understanding
VRD	Visually-Rich Document
QA	Question Answering
DocVQA	Document Visual Question Answering
VLM	Vision-Language Model
LLM	Large Language Model
MLLM	Multimodal Large Language Model
RAG	Retrieval-Augmented Generation
SFT	Supervised Fine-Tuning
UI	User Interface

1 Introduction

The integration of artificial intelligence into productivity and educational tools has transformed the way users interact with digital content. Among the most prominent advancements are generative models such as GPT [1], that demonstrate exceptional capabilities in understanding and processing natural language. These developments offer new opportunities for enhancing various applications, including note-taking systems, fundamental for students, researchers, and professionals in recording, organizing, and retrieving information. Leveraging AI in note-taking applications can significantly improve efficiency and effectiveness by automating content structuring and enhancing data interaction.

1.1 Motivation

Despite the rapid advancements in AI technologies, particularly in the development of large language models (LLMs) and multimodal models, current solutions primarily focus on providing basic text input functionalities without leveraging the full potential of AI for content analysis and knowledge extraction. Furthermore, support for incorporating AI capabilities into the note-taking process is limited, which could significantly enhancing user productivity by automating content generation and information retrieval. These gaps highlight the need for a more advanced system that not only handles various types of inputs but also provides seamless, proactive interaction with users.

First, most note-taking applications provide limited AI capabilities, which is largely due to the limitations of the underlying AI models. General-purpose large language models (LLMs), such as GPT [1], Qwen-VL [44], LLaMA [12], and InternVL [8], are optimized for high-level multimodal understanding and reasoning, making them less efficient for OCR-specific tasks that require detailed layout understanding and precise text extraction [52]. These models perform well in structured document analysis and multimodal reasoning but fall short in accurately recognizing and extracting information from dense, text-heavy documents. On the other hand, specialized LLMs for document tasks, such as Donut [20], Nougat [6], Vary [45], and GOT [46], are designed to handle optical character recognition (OCR) and document understanding more effectively but lack the comprehensive document question answering (DocQA) abilities of general-purpose models. These specialized models focus primarily on text extraction and layout parsing, limiting their usefulness for tasks that require deep semantic understanding and complex

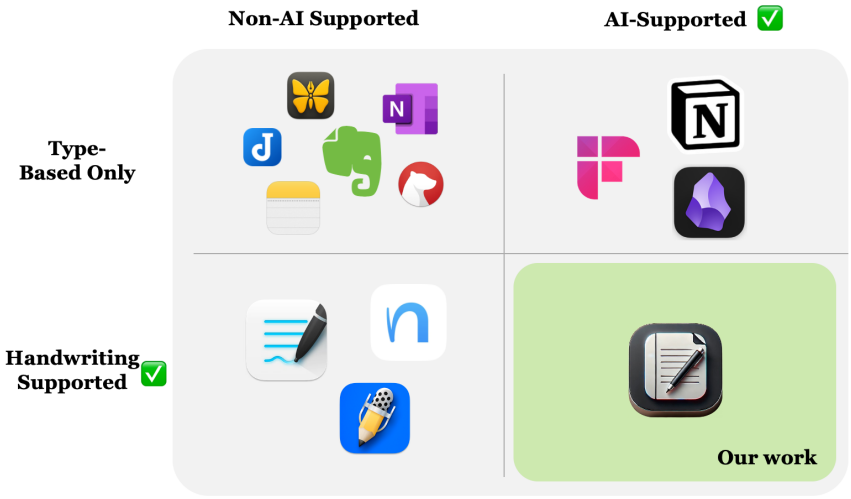


Figure 1: Comparison of Note-Taking Applications: AI Capabilities and Handwriting Support[2]

question-answering capabilities. Overall, there is no perfect model for all tasks under note-taking scenarios, as different models excel in specific areas but fall short in others.

Second, many note-taking applications lack effective handwriting support, limiting usability for users who prefer handwritten notes. Such support involves not only accepting handwritten input but also converting it into structured, searchable digital formats. This capability is essential for making handwritten notes easily searchable, significantly enhancing the ability to quickly locate specific information. The absence of these features reduces the overall functionality and user experience for those relying on handwritten notes.

Our preliminary investigation, as illustrated in Figure 1, shows that existing note-taking applications typically focus on either AI capabilities (top-right) or handwriting support (bottom-left), but not both simultaneously. This gap indicates the need for a unified solution that integrates advanced AI features with handwriting functionality, which our work aims to address.

[2]Icon images are obtained from https://icons8.com/

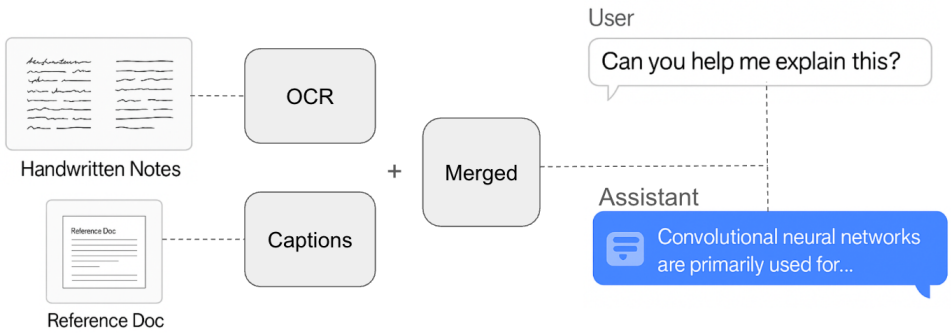


Figure 2: Handwritten notes and reference documents are digitized through OCR and caption extraction, then merged to power an AI assistant that answers user queries

1.2 Objectives and Deliverables

This project aims to develop an intelligent note-taking application that seamlessly integrates handwriting support with advanced artificial intelligence capabilities, creating a more structured and interactive note management experience. As illustrated in the diagram 2, the application will process user input through two main stages: first, converting handwritten notes and reference documents into digital format using OCR capabilities; then, using RAG technology to enhance QA functions, providing insightful answers to users' queries.

To achieve those functionalities, we will adopt a multimodal language modeling approach capable of handling various input types, including text, handwritten notes, tables, and formulas. This may involve a unified model or a composition of specialized modules for tasks such as OCR and question answering. These models will be trained to extract relevant information and enable context-aware interaction. The system consists of two major components: (a) an input processing pipeline that digitizes diverse document content into structured formats such as JSON or Markdown, and (b) an interactive question-answering module that enables users to pose context-specific queries via positional prompts. Additionally, we will develop a user-facing application that integrates these modules, supporting real-time interaction and AI-assisted note generation.

In summary, the main deliverables of this project include:

- **A Cross-Platform User Interface**
  - Develop a user-friendly and adaptive interface compatible with both desktop and mobile platforms.
  - Enable input flexibility by supporting typed text, handwritten content, and graphical elements such as sketches and diagrams.
- **A Structured Note Management System**
  - Transform unstructured notes, especially handwritten inputs, into indexed and searchable digital formats.
  - Introduce smart organization tools such as semantic search and automatic categorization to enhance information retrieval.
- **Integration with Multimodal Large Language Models**
  - Leverage multimodal LLMs to perform OCR and extract content from handwritten notes and supporting documents.
  - Facilitate natural language querying of notes, enabling intelligent and context-aware responses.
  - Construct and utilize a domain-specific dataset tailored for note-taking tasks to fine-tune the underlying models.

1.3 Contributions

This project applies VLM technologies to note-taking applications, presenting an innovative integrated solution that effectively combines OCR and QA functionalities. The accuracy and speed of OCR have been enhanced, enabling users to swiftly transform handwritten and printed text into editable digital content. In addition, leveraging the powerful language understanding and generation capabilities of LLMs has led to the development of a precise QA function. This allows users to obtain key information from their notes quickly through natural language queries, thereby improving the efficiency of knowledge acquisition.

An intelligent note-taking application that supports both handwriting and AI functionalities has been developed, offering users more choices in note-taking tools for office and learning scenarios. This application not only facilitates the input and editing of handwritten notes but also provides convenient features for organizing and managing notes. As a result, users can record, arrange, and utilize their notes more effectively. It is expected to find extensive applications in fields such as education and office work, providing users with more efficient and intelligent note-taking solutions.

Through an in-depth analysis of existing note-taking applications, this study has identified their shortcomings in AI functionality integration and proposed corresponding solutions. These findings offer a valuable reference for future note-taking application development.

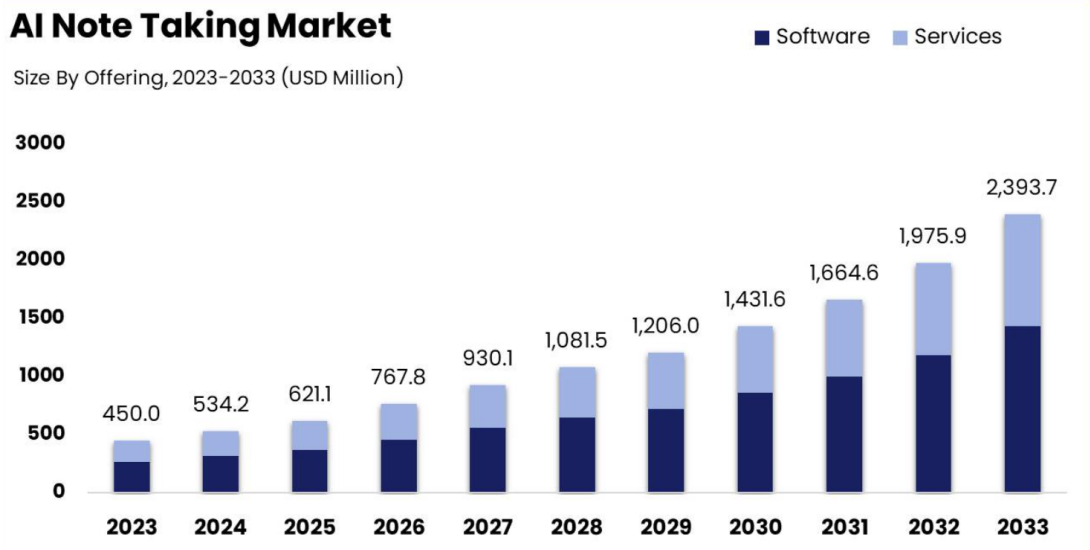


Figure 3: The Global AI Note Taking Market Apporximation from 2024 - 2033

2 Project Background

2.1 Background

Note-taking has evolved significantly from traditional pen-and-paper methods to digital solutions, reflecting broader technological advancements. In the early 2000s, basic text editors and word processors dominated digital note-taking. The 2010s witnessed the emergence of specialized note-taking applications with improved organization features, cloud synchronization, and cross-platform accessibility. By the 2020s, we've entered an era of intelligent note-taking, where AI capabilities are becoming increasingly integral to these applications.

The note-taking application market has experienced substantial growth, with projected estimates reaching $1.2 billion by 2029 from $450 million in 2023 shown in figure 3. This growth reflects both increasing digitalization across educational and professional sectors and rising awareness of efficient information management as a productivity enhancer.

The market currently features several distinct categories of applications:

Handwriting-focused applications (e.g., Notability, GoodNotes): These applications excel at digital handwriting, offering features like palm rejection, pressure sensitivity,

and various writing tools. They typically provide good document organization and basic search capabilities but lack advanced AI features.

Text-based productivity platforms (e.g., Notion, Evernote): These platforms emphasize structured content with robust organizational features and collaboration tools. Recent iterations have incorporated AI capabilities for content generation and organization, but most still have limited support for handwritten input.

Voice-to-text applications (e.g., Otter.ai): These specialized tools focus on transcribing spoken content into searchable text, often using AI for speaker identification and automated summarization.

The evolution of OCR technology represents a remarkable journey from rudimentary pattern matching systems to sophisticated neural network-based approaches. Early OCR implementations struggled significantly with varied handwriting styles and required meticulously formatted input, limiting their practical applications. Contemporary OCR systems, however, leverage advanced deep learning techniques to achieve unprecedented accuracy rates across diverse handwriting styles and document formats. Particularly transformative have been recent innovations in transformer-based architectures, which enable substantially improved context understanding and handwritten text recognition accuracy by processing entire sequences rather than isolated characters; few-shot learning approaches that allow systems to rapidly adapt to new handwriting styles with minimal examples, dramatically improving personalization capabilities; and strategic integration with language models that utilize linguistic context to resolve ambiguities and correct recognition errors, significantly enhancing overall system performance.

VLM have emerged as a groundbreaking technological development by seamlessly combining computer vision capabilities with sophisticated natural language understanding. Unlike conventional language models limited to textual input, VLMs can process and interpret both text and visual information simultaneously, making them exceptionally well-suited for complex document understanding tasks. Key technological breakthroughs underpinning modern VLMs include multimodal embedding spaces that enable fluid transitions between visual and textual information within a unified representational framework; cross-modal attention mechanisms that allow models to dynamically focus on relevant image regions when processing queries or generating responses; and advanced

document layout understanding capabilities that recognize and interpret structural elements beyond mere text content, including spatial relationships, formatting conventions, and visual hierarchies. State-of-the-art VLMs such as GPT-4o [33], Claude [9], and Qwen [3] have demonstrated remarkable proficiency in handling sophisticated document-related tasks, from extracting structured information from complex documents to answering nuanced queries about visual content, establishing a powerful technological foundation for next-generation note-taking applications that can truly understand and interact with multimodal content.

The convergence of advanced OCR technologies and sophisticated visual language models presents unprecedented opportunities for the next generation of note-taking applications. As these technologies continue to mature, we anticipate the emergence of truly intelligent note-taking systems capable of understanding, organizing, and retrieving multimodal information with human-like comprehension. Future note-taking applications will likely transcend current categorical boundaries, offering seamless integration of handwritten input, structured text, voice transcription, and visual content within unified, AI-enhanced platforms. These developments promise to fundamentally transform how information is captured, processed, and utilized across educational, professional, and personal contexts. The evolution from simple digital transcription tools to context-aware, multimodal knowledge systems represents not merely a technological advancement but a fundamental shift in human-information interaction paradigms.

2.2 Document Parsing Approaches

Document parsing and understanding have undergone significant transformations, evolving from traditional rule-based systems to sophisticated deep learning models as shown in Figure 4. This chapter provides a comprehensive review of the progression in document parsing methodologies (Section 2.2), encompassing traditional OCR techniques, deep learning-based recognition methods, and recent advancements in vision-language models. Additionally, it explores developments in document question answering systems (Section 2.3) and surveys pertinent datasets (Section 2.4) that have facilitated research and evaluation in this domain.

Traditional OCR. Traditional Optical Character Recognition (OCR) methods have evolved significantly through several stages before the advent of deep learning-based techniques.

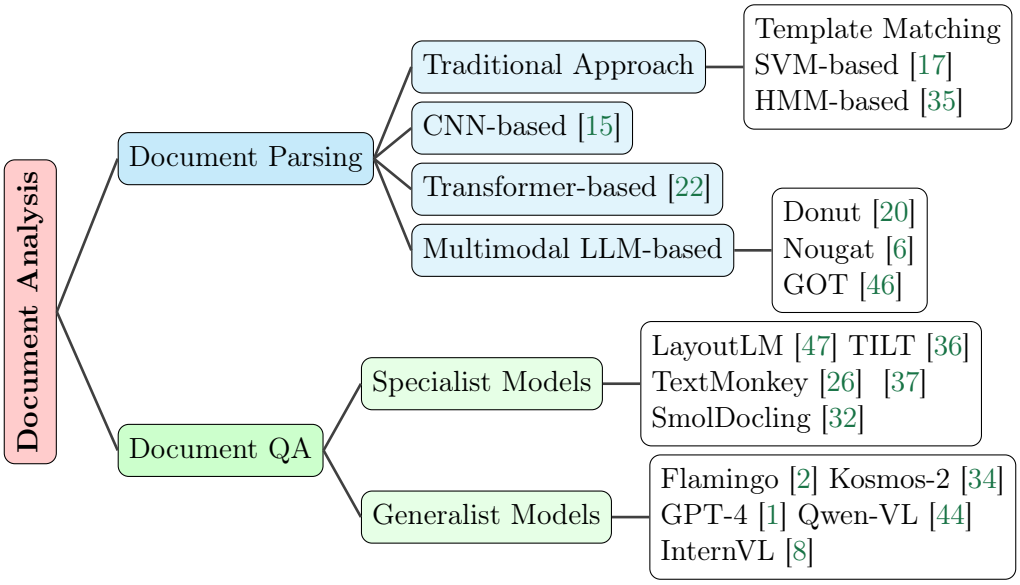


Figure 4: Taxonomy of Document Parsing and Question Answering Approaches

Initially, OCR systems relied primarily on classical computational algorithms [16], starting with simple template matching approaches. In these methods, recognition involved directly comparing input characters with predefined character templates, making them straightforward yet sensitive to font variations, scale changes, and noise. To address these limitations, feature extraction techniques [42] became prevalent, transforming characters into invariant shape descriptors, such as stroke patterns, gradient histograms, or contour-based features. Classification of these extracted features was commonly performed using statistical pattern recognition algorithms [17], including k-nearest neighbors (k-NN) and Support Vector Machines (SVMs), enhancing robustness to variations in input data. Additionally, statistical modeling approaches [35], notably Hidden Markov Models (HMMs), gained prominence by modeling text as a sequence of observations produced by underlying hidden character states. This probabilistic method effectively integrated spatial or contextual dependencies between characters, proving particularly successful in handwriting recognition scenarios. Collectively, these classical OCR methodologies established foundational paradigms that significantly contributed to the development of robust text recognition systems prior to the rise of more powerful Convolutional Neural Networks (CNNs) and Transformer-based architectures.

Deep Learning Based Recognition. Deep learning has fundamentally transformed OCR for both scanned documents and scene text images over the past decade [19]. A key advance was the adoption of convolutional neural networks (CNN) for visual feature extraction, replacing hand-crafted features and character segmentation pipelines with end-to-end trainable models [19]. Building on CNN backbones, subsequent methods introduced sequence modeling to handle variable-length text output. Specifically, recurrent neural networks with Connectionist Temporal Classification (CTC) enabled segmentation-free text recognition using word-level training only [15], while attention-based encoder–decoder frameworks brought a sequence-to-sequence paradigm that learns to align and transcribe characters in an autoregressive manner [29]. More recently, early Transformer-based architectures have eliminated recurrent modules in favor of self-attention mechanisms, achieving improved parallelism and long-range dependency modeling in text recognition [22]. These algorithmic shifts marked a progression from character-level classification toward holistic, end-to-end text-reading systems.

Recent Advancement with VLM. Recent advancements in Vision-Language Models (VLMs) have significantly improved document parsing by leveraging joint visual and linguistic understanding within unified modeling frameworks. Unlike traditional OCR and early deep learning-based approaches that primarily focus on text transcription, VLMs integrate multimodal information, allowing simultaneous recognition, comprehension, and semantic interpretation of document contents. Models like Donut [20, 21] eliminate the dependency on OCR engines by directly processing document images through transformer architectures, achieving end-to-end document understanding. Similarly, Nougat [6] focuses on academic documents, effectively converting complex layouts and mathematical expressions into structured markup. GOT [46] proposes a unified end-to-end model capable of handling diverse OCR tasks, including the recognition of plain text, formulas, tables, and charts, by employing a high-compression encoder and a long-context decoder. These models demonstrate enhanced capabilities in recognizing handwritten text, distorted or partially obscured text regions, and diverse layouts that often challenge previous methodologies. Furthermore, the multimodal pretraining paradigm empowers these models to generalize effectively across varied document domains, reducing the dependence on extensive annotated data specific to the target application. Consequently, VLMs represent a significant advancement toward general-purpose, end-to-end solutions capable of parsing diverse and semantically rich document images.

2.3 Document Question Answering

Document Question Answering (DocQA) has evolved rapidly with the rise of vision-language models. Early approaches extended visual QA models to read text by incorporating OCR outputs as an additional modality. For example, models like LoRRA [39] attended to text regions in images and learned to either copy recognized strings or infer answers from them. More recent Transformer-based architectures enabled a more unified encoding of document content: LayoutLM [47] and related models jointly learn text, layout, and visual features by pre-training on large-scale document corpora, substantially improving document understanding performance. These models leverage multi-modal self-attention to handle the dense layout, allowing them to reason over long sequences of tokens and visually rich elements. However, their QA approach is typically extractive—answers are selected from the text—which can be limiting in cases requiring generation or external reasoning. To address this, generative transformer models have been introduced. TILT [36] is one such example that uses an encoder-decoder to generate answers from text and image features, enabling flexible responses beyond copying text. Another challenge is handling the long context of multi-page documents. Recent works propose hierarchical transformers and retrieval-augmented models that process pages in segments and aggregate relevant information, allowing QA over entire documents without exceeding memory limits [36]. Finally, large-scale multimodal language models push DocQA further by marrying powerful visual encoders with large language models. Systems like Flamingo [2] are trained on massive image-text corpora and can directly ingest document images alongside text, performing open-ended question answering through unified attention over both modalities. Such models exploit multimodal pre-training to learn aligned visual and textual representations, demonstrating robust reasoning on document images and often achieving state-of-the-art DocQA results.

2.4 Related Datasets and Metrics

OCR Datasets. OCR datasets are fundamental for training and evaluating models across various text recognition tasks, encompassing scene text, document parsing, and handwriting recognition. TextOCR provides approximately 900,000 high-quality word-level annotations on natural images, facilitating research in arbitrary-shaped scene text detection and recognition [38]. FUNSD offers a collection of 199 fully annotated scanned

Table 1: Representative Datasets for Document Parsing, Handwriting OCR, and Visual Question Answering

Task	Datasets
Document Parsing	TextOCR [38], FUNSD [18], SynthDoG-en [20]
Handwriting OCR	IAM [27], HME100K [50], CASIA-HWDB [23]
Visual Question Answering	OCR-VQA [30], TextVQA [39], ST-VQA [5], ScreenQA [51], SlideVQA [40], PDF-VQA [11], VQA-CD [30], VQAonBD [4], DocVQA [28], UReader QA [49]

forms, supporting tasks such as text detection, OCR, spatial layout analysis, and form understanding in noisy scanned documents [18]. To augment training data, SynthDoG-en generates synthetic English documents, enabling OCR-free models like Donut to learn from diverse and realistic document layouts [20].

In handwriting recognition, the IAM Handwriting Database comprises forms of handwritten English text, segmented at the word and line levels, and is extensively used for training and evaluating handwriting recognition systems [27]. For handwritten mathematical expressions, HME100K offers a large-scale dataset with 100,000 images collected from thousands of writers, capturing the complexity of mathematical notation in handwritten form [50]. In the context of Chinese handwriting, CASIA-HWDB provides a vast collection of offline handwritten Chinese characters, encompassing over 3.9 million samples across 7,356 classes, serving as a fundamental resource for Chinese character recognition research [23].

For evaluating OCR tasks, several comprehensive benchmarks have been developed to assess the capabilities of models across diverse scenarios and challenges. Notably, OCRBench [25, 13] and CC-OCR [48] stand out for their extensive coverage and rigorous evaluation metrics.

DocQA Datasets. Document Question Answering (DocQA) has seen significant advancements with the introduction of diverse datasets tailored to various document types and question-answering challenges. OCR-VQA focuses on answering questions by reading text in images, providing a dataset that emphasizes the integration of OCR and VQA tasks [30]. TextVQA requires models to read and reason about text in images to answer

questions, highlighting the necessity of incorporating text understanding in VQA models [39]. SlideVQA offers a multi-image document VQA dataset containing slide decks, requiring complex reasoning including single-hop, multi-hop, and numerical reasoning [40]. PDF-VQA provides a dataset for real-world VQA on PDF documents, examining document understanding from various aspects [11]. VQA-CD and VQAonBD are datasets focusing on business documents, with VQAonBD being part of the ICDAR 2023 competition [4]. DocVQA consists of 50,000 questions defined on over 12,000 document images, serving as a standard benchmark for document VQA [28]. UReader QA introduces an instruction-tuning dataset covering various domains of visually-situated language understanding, including documents, tables, charts, natural images, and webpages [49].

2.5 Modern Application Development

The landscape of web application development has evolved significantly in recent years, with a growing emphasis on creating seamless cross-platform experiences while maintaining developer productivity. Electron has emerged as a leading framework for developing cross-platform desktop applications using web technologies. Introduced by GitHub in 2013, Electron combines Chromium's rendering engine with Node.js runtime, enabling developers to build desktop applications using HTML, CSS, and JavaScript. A key advantage of Electron is its ability to access native operating system features through JavaScript APIs, bridging the gap between web and native capabilities.

Flask has established itself as a lightweight yet powerful Python web framework ideal for building APIs and backend services. The microframework approach allows developers to include only necessary components, resulting in leaner backend services that can be deployed in various environments from local development to cloud platforms. This pattern allows developers to leverage Python's extensive libraries for natural language processing, machine learning, and data analysis while providing a responsive user interface through modern web technologies.

3 Methodology

This methodology shown in fig. 5 presents an end-to-end framework for building a VLM-powered note-taking assistant, organised into two parallel tracks: Model Development and App Development.

Model Development covers everything required to train the language model. It starts with collecting and preprocessing a tailored dataset, continues with supervised fine-tuning that adapts a base LLM to note-assistant tasks, and finishes with rigorous evaluation before the model is deployed.

App Development focuses on the user-facing system that connects to the LLM. The backend implements an API gateway and data-processing pipelines, while the frontend offers an intuitive interface that includes an interactive canvas for rich note-taking, handwriting recognition, and robust note-management features. Together, these tracks deliver a cohesive, user-centric application powered by finetuneing our own VLM(s).

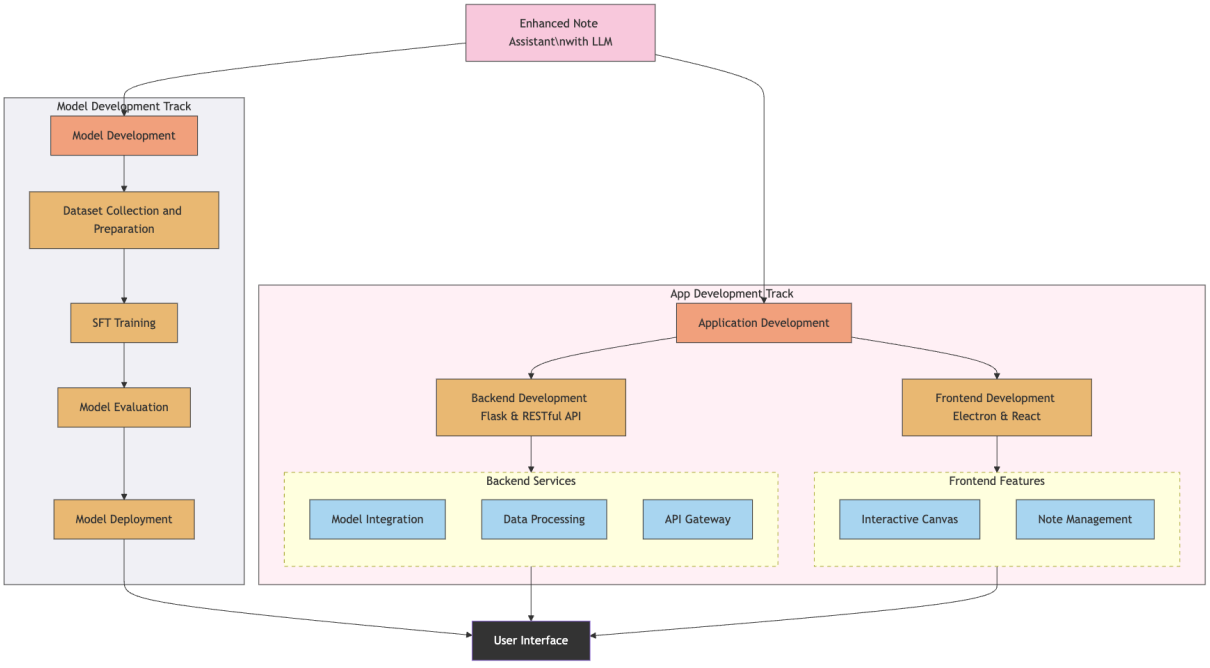


Figure 5: Methodology Overview.

3.1 Overall Model Design

The workflow illustrated in fig. 6 starts with a document input that may combine unstructured text, structured form fields, and page element such as tables or figures. This raw data first passes through a unified feature-extraction block that fuses textual and visual cues. The resulting representation is forwarded to a hidden-state generator, producing a dense vector embedding of the entire document. Task-specific decoder heads are then attached—for example, those for question answering in document understanding, element detection and localisation, or document indexing and search. By sharing a single encoder while learning specialised output layers, the architecture supports a broad set of document-processing objectives, promotes transfer learning, and minimises overall complexity.

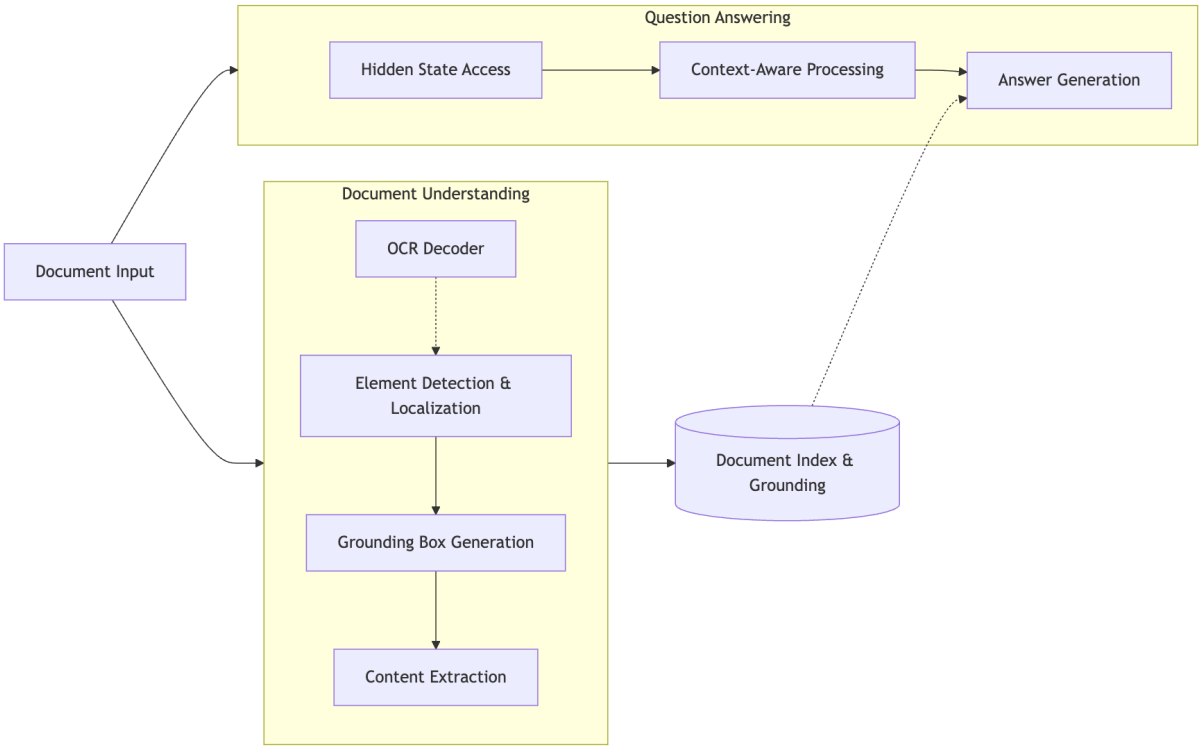


Figure 6: The designed architecture of model workflow

3.2 Document Parsing and VQA Model

Model Architecture. Our document parsing model follows a standard self-attention paradigm common in vision-language models. The overall architecture can be divided into three primary components:

- **Image Encoder**: A visual backbone that extracts rich feature representations from document images. This component processes the raw pixels and generates high-dimensional embeddings that capture both local and global visual patterns in the input document.
- **MLP Projector**: A projection layer that maps the visual features into the same embedding space as the text tokens. This alignment of visual and textual representations is critical for effective cross-modal understanding.
- **Language Decoder**: A transformer-based decoder that generates textual outputs based on the combined visual and textual inputs. This component handles the sequential prediction of tokens for various document understanding tasks.

Based on extensive ablation studies (detailed in Section 4), we selected SigLip as our image encoder and Qwen2-0.5B as our language decoder, finding this combination to provide the optimal balance between performance and efficiency.

Following established practices in multimodal modeling, we concatenate the system prompt, image tokens, and user-defined prompt into a single sequence and perform next-token prediction to generate the answer string. This task can be formally modeled as:

$$P(y|I,x) = \prod_{i=1}^{n} P(y_i|y_{<i}, \mathbf{E}(I), x) \tag{1}$$

where I represents the input image, x is the text prompt, y is the generated answer, $\mathbf{E}(I)$ denotes the image encoding process, and $P(y_i|y_{<i}, \mathbf{E}(I), x)$ is the probability of generating the i-th token given the previous tokens, image features, and input prompt.

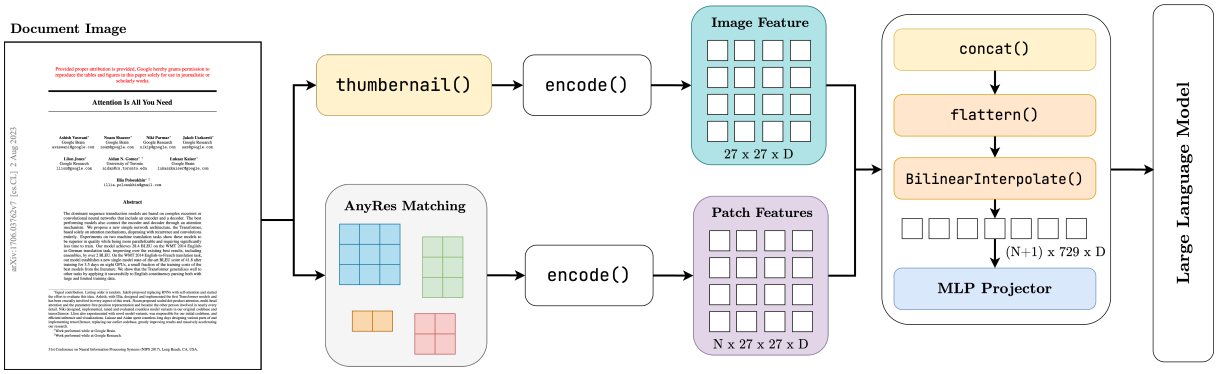


Figure 7: Image Preprocessing and Visual Representation.

Visual Representations. For efficient processing of diverse document images, our approach to visual representation follows established methods in multimodal models while introducing specific adaptations for document understanding.

Current large models typically process images by splitting them into patches and encoding each sub-image separately to accommodate varying resolutions. To maintain global context information, thumbnails are often added before these sub-images to help with localization. We adopt this approach for our visual encoding component.

In our application, users frequently input images with vastly different sizes and aspect ratios (such as presentation slides or elongated handwritten canvas). To handle this variety of document images, we adapted the method from LLaVA-Next, allowing dynamic matching of images to predefined resolution tiers. This matching process considers the maximum effective pixel utilization to select the optimal resolution, then applies linear interpolation to ensure the number of images doesn't exceed the maximum value. One difference with the original method is that we replaced the spatial unpadding operation with resizing, as unpadding can negatively impact the model's visual grounding capabilities according to our experiment. For our settings, each sub-image produces 729 image tokens. Therefore, the final number of image tokens is calculated as:

$$\#\text{image\_token} = \begin{cases} (N+1) \times 729, & \text{if } (N+1) \times 729 \leq \text{max\_tokens} \\ \text{max\_tokens}, & \text{otherwise} \end{cases} \tag{2}$$

where N is the number of sub-images after partitioning the input document.

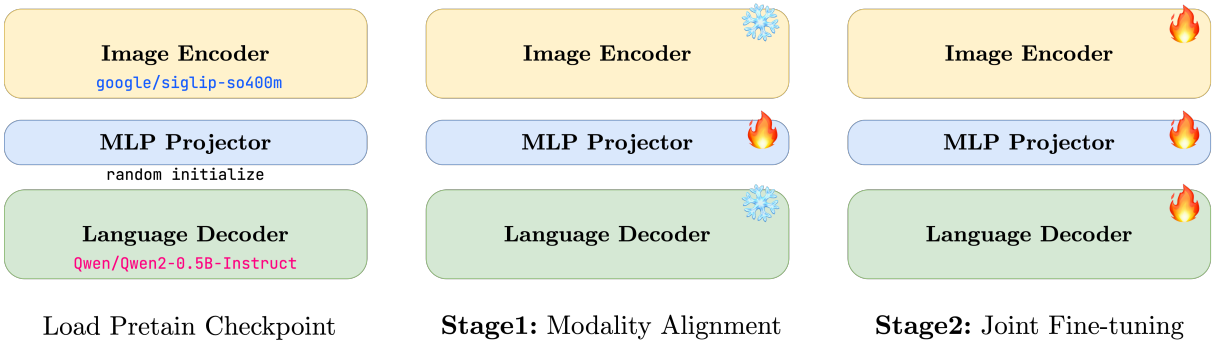


Figure 8: Two-stage fine-tuning workflow. flame and snowflake represent the activation and freezing of the corresponding model components. The first stage focuses on training the MLP projector, while the second stage fine-tunes the entire model end-to-end.

Model Fine-tuning. Our fine-tuning strategy follows a two-stage approach to effectively align visual and textual modalities while preserving the pre-trained knowledge in both encoders.

For our model, we utilize `google/siglip-so400m-patch14-384` as our visual encoder and `Qwen/Qwen2-0.5B-Instruct` as our language decoder. These pre-trained models provide strong foundations in their respective modalities, but require careful alignment to work effectively together on document understanding tasks.

- **Stage 1: Modality Alignment:** In the first stage, we focus exclusively on training the MLP projector while keeping both the image encoder and language decoder frozen. This approach allows the model learn the mapping between the two pre-trained embedding spaces without disturbing the specialized knowledge each model has acquired during pre-training. This stage also requires significantly less computational resources than full model fine-tuning, allowing for rapid experimentation with different configurations, while keeping the pre-trained weights frozen preserves the knowledge encoded in both models while establishing a bridge between them.
- **Stage 2: Joint Fine-tuning:** In the second stage, we unfreeze all components—the image encoder, MLP projector, and language decoder—and fine-tune the entire model end-to-end. This comprehensive fine-tuning allows the model to learn the intricate relationships between visual and textual modalities, enabling it to generate more accurate and contextually relevant outputs.

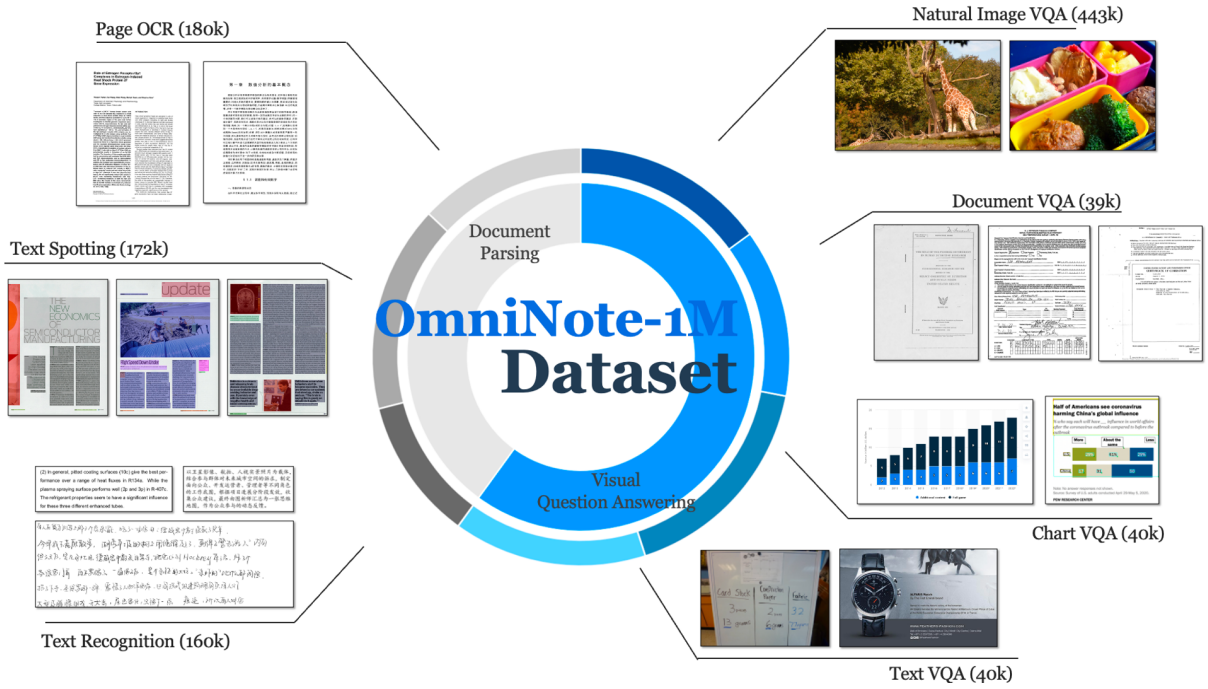


Figure 9: Dataset Overview of OmniNote-1M

We use AdamW as our optimizer with a learning rate of (2e-5) and (1e-5) for each stage and global batch size of 64. This two-stage approach allows us to leverage the strengths of both pre-trained components while effectively bridging the semantic gap between visual document features and textual understanding.

This progressive fine-tuning approach allows our model to effectively bridge the semantic gap between visual document features and textual understanding while preserving the strengths of both pre-trained components.

3.3 Data

Dataset Overview. To train and evaluate the selected models effectively, a diverse and extensive dataset was prepared, incorporating both real and synthetic data. The OmniNote-1M dataset was designed to align with our project objectives of enhanced information extraction and multimodal capabilities. It comprises 180k samples for Page OCR, 172k samples for Text Spotting, and 160k samples for Text Recognition, facilitating robust document understanding. Additionally, the dataset includes tasks related to Visual Question Answering, which are categorized as follows: 443k samples for Natural Image

VQA, 39k samples for Document VQA, and 40k samples for Chart VQA. Together, these datasets cover a diverse array of scenarios, ranging from structured document parsing to unstructured visual input.

Table 2: Dataset statistics of *OmniNote-1M*.

Task	Name	Dataset Type	Source	Image type	Subtask	Qty
Document Parsing	OmniNote-OCR	self-collected	arXiv (EN) CC-MAIN-PDF (EN) ebooks (CN)	Digital Document Scanned Document	Page OCR	180k
					Text Spotting (w/ grounding)	172k
					Text Recognition (Text/Equation/Table)	157k
	OmniNote-HWOCR	synthesized (from public)	CASIA-HWDB (CN) IAM-line (EN)	Scanned Handwriting (Offline)	Text Recognition	3k
			/	Scanned Document Scanned Handwriting	Page OCR	10k
Visual Question Answering	OmniNote-VQA	public dataset	VQA v2	Natural Image	/	443k
			LRV_Chart	Natural Image		7k
			TallyQA	Natural Image		38k
			TextVQA	Digital Image Captured Image Scene Text Image		34k
			DocVQA	Scanned Document		39k

The OmniNote-1M dataset was constructed using a combination of public datasets, synthetic data, and self-collected data. The document parsing subset consists of self-collected data from sources such as arXiv and Common Crawl, which are used for Page OCR and Text Spotting. The Text Recognition task also includes synthesized handwritten data, sourced from public datasets like CASIA-HWDB [23] (for Chinese) and IAM [27] (for English). These handwritten datasets are specifically designed for single-line recognition. For our use case, multiple single-line instances were randomly stacked together to create a paragraph-level images, simulating handwritten notes. The VQA tasks leverage public datasets such as VQA v2 [14], LRV-Chart [24], and TextVQA [39], ensuring a broad range of natural and document image types. This diverse combination of datasets supports both structured document tasks and general VQA challenges.

Data Collection. The OmniNote-OCR subset was primarily collected through web crawling from multiple sources, creating a comprehensive document corpus. Web scraping tools were deployed to gather data from Common Crawl, e-book platforms, ChinaXiv, and other repositories containing PDFs of academic papers, books, textbooks, and presentation slides—document types commonly used in note-taking scenarios. This subset includes 723,212 PDF pages in total, providing substantial data for model training. For this portion of the dataset, we utilized open-source annotation pipelines such as Marker and MinerU to label the data effectively. Additionally, we crawled a substantial number of academic papers (∼60k PDF pages) from arXiv to enhance the dataset's diversity. For these arXiv papers, we followed the data preprocessing methodology established in the Nougat [6], converting the original LaTeX (tex) files into Markdown format to serve as ground truth labels. This approach allowed us to capture the rich structural elements of academic documents while maintaining a standardized annotation format suitable for model training.

Data Synthesis. To enhance the dataset, we generated synthetic data using existing OCR character sets and data synthesizing pipeline like SynthDoG [20]. This process involved creating a variety of document types, including academic papers, textbooks, and presentation slides, to ensure a comprehensive representation of different content formats. The synthetic data generation process was designed to mimic real-world scenarios, incorporating various elements such as text, images, tables, and charts. By synthesizing this data, we aimed to fill gaps in the existing dataset and provide the model with diverse examples for training. The generated data was then integrated into the OmniNote-1M dataset, enriching its content and improving the model's ability to handle a wide range of note-taking tasks. (See fig. 16 and fig. 17 in Appendix for synthesized samples)

Dataset Preprocessing. To ensure the high quality and diversity of the OmniNote-1M dataset, a rigorous filtering pipeline was applied to the self-collected data. For OCR and VDU tasks, diversity means the model needs to adapt to various document types and different layout structures, so our first step in data preprocessing was to filter data based on document visual features.

To extract the visual feature from PDF document, we use a document analysis model based on YOLO-v10 [43], which was fine-tuned for document layout feature extraction on

various document types. This allowed us to effectively analyze the structural elements of each document, such as text blocks, images, and tables, in a high-dimensional space. Next, we use K-nearest Neighbors to group documents with similar layouts and structures. This clustering process enabled us to divide the original dataset of over 780k raw samples into distinct clusters based on layout patterns. As a result, 180k high-quality samples were selected that were both structurally diverse and representative of the broader dataset. This careful selection ensures that the data used for model training is clean, well-organized, and balanced across various document types.

Table 3: Dataset Format Design for Text, Table, and Equation Tasks.

Task	Format	Sample
General OCR	text	Here is some sample text.
OCR with format	markdown	#Title\n\n ##Section\n\n Some sample text.
Equation	LaTeX	\[\begin{array}{r}{n=64.} ...
Table	html	table><tr><td>Compound</td>...</table>

Table 4: Dataset Prompt Design for OCR, VQA Tasks with Grounding Labels.

Task	Prompt	Label
General OCR	OCR:	<text>***OCR Result***</text>
Text Spotting (Paragraph Level)	OCR with grounding:	<text>***OCR Result***</text><box>x1 y1 x2 y2</box> (repeated for N page elements)
Text Recognition	Read the text in the <ref>box</ref><box>x1 y1 x2 y2</box>.	<text>***OCR Result***</text>
VQA	***Question***. Answer:	***Answer***.
VQA with Grounding	***Question***. Provide the location coordinates of the answer when answering the question. Answer:	***Answer***. <box>x1 y1 x2 y2</box>

Data Format and Prompt Design. For standardization purposes, the dataset was formatted according to the specific requirements of each task as shown in table 3 . For general OCR tasks, we utilize plain text for its simplicity and universal compatibility. For OCR with formatting tasks, we adopted specialized formats: markdown for documents, LaTeX for equations, and HTML for tables. These format choices were deliberate and strategic. Markdown provides a lightweight yet expressive syntax for document structure, preserving

headings, lists, and emphasis while maintaining human readability. LaTeX was selected for mathematical equations due to its precision in representing complex mathematical notations, symbols, and structures with unambiguous semantic meaning—an essential feature for accurate equation interpretation and rendering. HTML was chosen for tables because of its explicit row-column structure, which preserves the spatial relationships between data cells while enabling semantic annotations for headers and special cell types. This format diversity ensures that each content type is represented in its most appropriate and information-preserving format.

In addition, we followed and extended the prompt design methodology from a prior work, TextMonkey [26], while adopting special token settings from Qwen Tokenizer [44] as shown in table 4. The careful design of prompts is crucial for multimodal models, as it bridges the gap between image inputs and text outputs by providing clear instructions that guide the model's attention and response generation. Our prompt templates include consistent system messages that define the model's role and capabilities, followed by clearly structured instructions that specify the expected input-output behavior. For example, the VQA tasks utilize a clear question-answer format with explicit markers for questions and answers, while all grounding tasks provide positional coordinates normalized from 0 to 1000 to ensure consistent spatial reference regardless of original image dimensions, similar to the approach used in recent MLLMs [7].

Image Augmentation. To ensure the robustness of our model in real-world scenarios, image augmentation was essential for training with document images. When users capture notes through mobile phone cameras or scanners, the resulting images often exhibit variations in lighting, angle, and quality that differ from pristine training data. These practical challenges include uneven illumination, shadow effects, blurring from hand movement, and perspective distortions. By systematically applying targeted augmentations during training, we enhanced the model's capability to process imperfect document images encountered in practical applications. Our augmentation pipeline simulated these real-world conditions, creating a more diverse and representative training dataset that better prepared the model for handling the variability present in user-submitted content. (See fig. 18 in Appendix for visualized samples)

For our dataset, we implemented the following common transformations:

- **Gaussian Blur**: Simulates focus issues in camera captures by applying variable blur kernels ($\sigma = 0.5 - 1.5$), mimicking common smartphone photography challenges
- **Shadow**: Adds realistic shadow effects with adjustable intensity and direction to simulate uneven lighting conditions common in impromptu document scanning
- **Perspective Transformation**: Applies random geometric distortions ($\pm 10^{\circ}$ rotation, $\pm 5\%$ shearing) to account for non-perpendicular camera angles during document capture
- **Adaptive Binarization**: Converts grayscale images to binary format with locally adaptive thresholding, simulating high-contrast scanning outputs while preserving text legibility in varying lighting conditions

3.4 Application Development

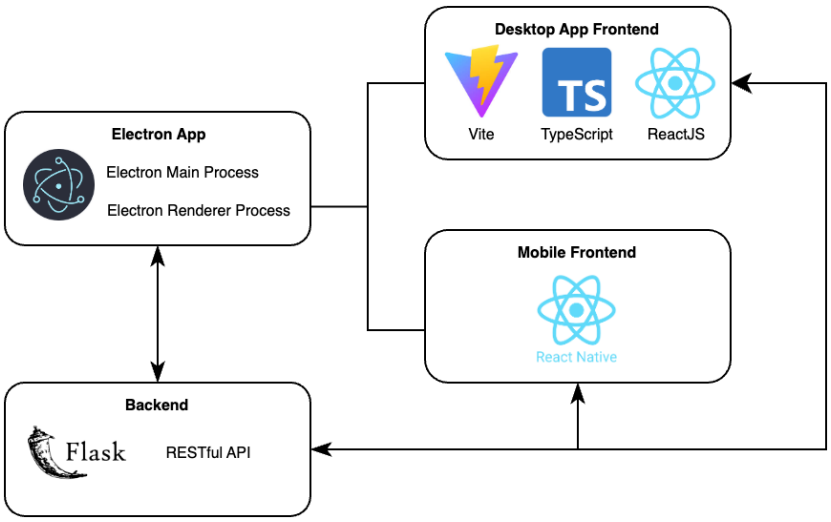


Figure 10: System Architecture of Our Application.

Building our note application focuses on three key tasks: designing the system architecture, developing both the front-end and back-end, and integrating essential functions such as note management alongside advanced features.

System Design. The application adopts a modular architecture that separates concerns across the back end and front end while emphasizing cross-platform support, high performance, and ease of maintenance. As depicted in fig. 10, an Electron front end communicates with a Flask back end, enabling seamless operation on multiple operating systems and devices.

During typical use, a desktop or mobile client sends RESTful HTTP requests from the front end to the Flask server. These requests carry user inputs—such as images, handwritten notes, or reference files—for OCR or note-retrieval tasks. The server processes the data and responds with JSON payloads, which the front end then renders to update the user interface.

Backend Design. The Flask-based back end hosts the core application logic and handles calls to external services through a structured RESTful API with three primary components:

- **Canvas Management**: Implements CRUD operations via `/api/canvas/` endpoints, storing canvas data as structured JSON with support for multi-modal elements including handwritten strokes (with pressure/tilt metadata), formatted text blocks, and positioned images
- **File Processing**: Handles document uploads through `/api/files/` endpoints, supporting PDF and image formats with automatic conversion to Markdown via Vision Language Models (VLM) that perform OCR and semantic analysis
- **Knowledge Interaction**: Powers contextual Q&A through `/api/qa/ask` endpoints using Retrieval Augmented Generation (RAG), indexing both original content and VLM-generated Markdown representations

Flask's lightweight architecture enables efficient routing for machine learning services, particularly optimized for stroke processing pipelines that preserve handwriting dynamics (position/pressure/brush parameters) while maintaining low-latency document conversion workflows. All user-generated content – including canvas states (`width/height`, `background_color`, `elements` hierarchy) and original files – persists in versioned storage with automatic `created_at`/`updated_at` tracking. The system orchestrates VLM conversions through parallel pipelines: one stream for visual content interpretation (diagrams,

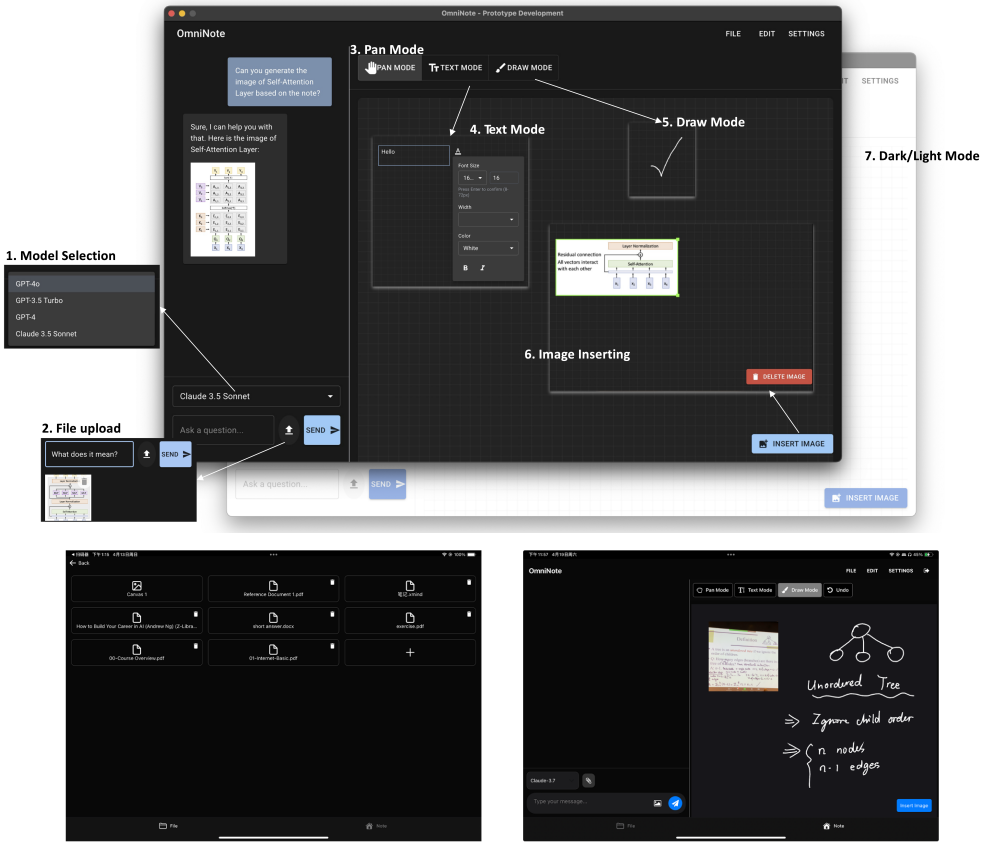
handwritten notes) and another for document structure extraction (PDF layouts, image text blocks), ensuring semantic coherence in the generated Markdown output.

Model Deployment. For deploying our fine-tuned document parsing model, we utilized SGLang, a specialized inference framework optimized for language models. This deployment strategy offered significant performance advantages compared to traditional deployment methods such as the standard Hugging Face Transformers library.

The SGLang framework enabled us to optimize the decoding process through specialized kernels and memory management techniques specifically designed for transformer-based models. Our benchmarks demonstrated that this approach accelerated the decoding phase by 12-15 times compared to HuggingFace's transformers implementation, without compromising the model's accuracy or output quality. This performance improvement was particularly critical for our note-taking application, where users expect real-time responses when processing document images. The acceleration allowed us to maintain interactive response times even when handling complex documents with multiple elements like tables, equations, and mixed text formats. Additionally, SGLang's batching capabilities improved throughput when handling concurrent requests, making the system more scalable for batch processing scenarios where users submit multiple documents simultaneously.

Frontend & UI Design. The front-end adopts a single code-base that runs seamlessly on both desktop and mobile devices. Its responsive interface lets users capture notes, organise them, and query the underlying models through an intuitive workflow. Real-time modules—such as on-device handwriting recognition and sketch-to-text conversion—are embedded directly in the UI, which has been designed with a clean, visually appealing layout to maximise usability. The cross-platform frontend not only offers a responsive, visually clean interface for note capture and model queries, but also adds two key capabilities:

- **File manager**: A file explorer lets users create new canvases or upload files into an organised folder tree, making it easy to browse, rename, and version documents.
- **Versatile canvas workspace**: Within each document, a rich canvas allows free-hand writing, typed text, shape stencils, and image insertion. These inputs can be edited,



(a) File Manager (upload file/creating canvas)

(b) Canvas Workspace

Figure 11: User-interface prototypes of the note-taking application on desktop (top) and mobile (bottom) platforms.

resized, or layered, providing a flexible surface for sketching ideas or composing structured notes.

Together, these modules complement the real-time handwriting recognition and sketch-to-text conversion already built into the UI, delivering a cohesive and highly interactive user experience across desktop and mobile devices. Figure 11 shows the UI design desktop application and the mobile version of the file manager and the canvas workspace.

4 Experiments and Results

This chapter presents comprehensive experiments and analyses of our note-taking system with a particular focus on the OCR component's performance. We evaluate both technical metrics and real-world usability to demonstrate the effectiveness of our approach. We conducted systematic experiments to assess various aspects of our system, from OCR accuracy to query response quality.

Our experiments reveal that smaller models (0.5B to 2B parameters) face significant challenges when handling diverse user queries in document question-answering (DocQA) tasks, despite their computational efficiency. The performance gap between these compact models and larger, more robust models becomes particularly evident when processing complex or ambiguous queries. To address this limitation, we conducted extensive testing comparing API-based solutions that leverage industry-leading large language models against our optimized approach on our specialized dataset. This comparative analysis provides valuable insights into the trade-offs between model size, inference speed, and response quality in the context of note-taking applications.

4.1 Dataset and Evaluation Metrics

Dataset. For OCR tasks, we combined following three benchmark datasets for evaluation:

- **OmniNote-Val**: Contains 700 pages from the same source as our training set, selected from the remaining pages after data filtering. This dataset encompasses various document types including academic papers, books, and slides.
- **OCRBench (document parsing subset)**: Includes various scanned and photographed document images containing diverse page elements such as formulas, tables, and other structured content.
- **CC-OCR**: Similar to OCRBench but provides more Chinese samples, handwritten samples, and partially degraded image data.

Metrics. For OCR tasks, we employed the following metrics to evaluate text recognition accuracy and bounding box quality:

- **Edit Distance**: This metric quantifies the difference between the predicted text and ground truth text by counting the minimum number of operations (insertions,

deletions, or substitutions) required to transform one string into another. A lower edit distance indicates better text recognition performance.

- **F1 Score**: Commonly used in object detection tasks, F1 score serves as an alternative to mean Average Precision (mAP) since large language models do not provide confidence scores in their outputs. It represents the harmonic mean of precision and recall, providing a balanced measure of detection accuracy. we used $IOU \geq 0.5$ as the threshold for true positive detection in following experiments.

For Document Question-Answering tasks, we evaluated models based on answer accuracy across different subsets of our test data. This metric measures the percentage of questions for which the model's response matches the ground truth answer, allowing us to assess the model's comprehension and reasoning capabilities when handling document-specific queries.

4.2 Model Implementation Details

Our OCR system is built on the foundation of the LLaVA architecture. The model implementation leverages the Huggingface transformers library, which provides a robust framework for implementing and fine-tuning multimodal models. The architecture follows the conventional encoder-decoder (single stream) structure, where the vision encoder processes image inputs and the language decoder generates textual outputs. We employed a systematic training approach with two distinct phases: Modality Alignment and Joint Fine-tuning. For optimization, we selected the AdamW optimizer paired with a linear warmup followed by cosine decay learning rate schedule. Different learning rates were applied across the training phases: 2e-5 for the Modality Alignment phase and 1e-5 for the Joint Fine-tuning phase. To balance computational efficiency with numerical stability, we trained using bf16 precision. The training infrastructure consisted of 16 NVIDIA A100 GPUs with a global batch size of 128, and we conducted training for 10 epochs over the entire training dataset. For handling visual and text data, we configured the maximum visual tokens to be 5 times the encoder patch size ($\sim$ 3600 tokens), while the maximum sequence length was set to 8192 tokens to accommodate document-length inputs.

Type	Model	Param	Text		Equation		Table	
			EN	CN	EN	CN	EN	CN
Pipeline Tools	Mathpix	/	<u>0.101</u>	0.358	<u>0.306</u>	**0.454**	0.322	0.416
Expert VLMs	Nougat	0.4B	0.365	-	0.488	-	0.622	-
	GOT	0.5B	0.191	0.314	0.360	0.523	0.459	0.520
	SmolDocling	0.3B	0.262	0.838	0.753	0.997	0.729	0.907
General VLMs	GPT-4o	/	0.144	0.409	0.425	0.606	0.363	0.474
	Gemini2.5 Flash	/	0.119	**0.147**	0.354	<u>0.473</u>	**0.193**	**0.206**
	Qwen2-VL	72B	0.252	0.251	0.468	0.572	0.591	0.587
	InternVL2	76B	0.353	0.290	0.543	0.701	0.616	0.638
	InternVL3	78B	0.116	<u>0.206</u>	0.380	0.553	<u>0.272</u>	<u>0.276</u>
Ours	SigLip-Qwen	0.5B	**0.081**	0.214	**0.277**	0.578	0.283	0.281

Table 5: Compare to SOTA Methods on our Test Sets. **Bold** and <u>Underlined</u> Scores Refer to Best or Second Best Model.

4.3 Comparison with SOTA Methods

Document Parsing Model. We conducted a comprehensive comparative analysis of our model against state-of-the-art OCR methods across different categories, as shown in Table 5. The evaluation focused on edit distance metrics for three content types (text, equations, and tables) in both English and Chinese languages, with lower scores indicating better performance.

Our analysis reveals several key findings. Despite having only 0.5B parameters, our SigLip-Qwen model achieves remarkable performance that rivals or exceeds much larger models with 70B+ parameters. This parameter efficiency is particularly notable when compared to general VLMs like Qwen2-VL (72B) and InternVL models (76-78B). In English text recognition, our model achieves the best performance with an edit distance of 0.081, outperforming all competitors including specialized commercial solutions like Mathpix (0.101) and significantly larger models like GPT-4o (0.144) and InternVL3 (0.116).

For English equations, our model also achieves the best performance with an edit distance of 0.277, demonstrating its particular strength in handling mathematical content despite its small size. This is especially impressive compared to specialized tools like Mathpix (0.306) and larger models like Gemini2.5 Flash (0.354). While our model doesn't lead in Chinese content recognition, it still delivers competitive performance (0.214 for

Model	$\textbf{ChartQA}_{\textbf{Test}}$	$\textbf{OCRVQA}_{\textbf{Test}}$	$\textbf{TextVQA}_{\textbf{Val}}$	$\textbf{DocVQA}_{\textbf{Test}}$
Qwen2VL-2B (Baseline)	71.48%	74.65%	78.67%	89.00%
Qwen2VL-7B (Baseline)	83.0%	-	84.3%	**94.5%**
Qwen2VL-2B-1M-10epoch (ours)	59.84%	43.37%	69.41%	77.16%
Qwen2VL-2B-80k-10epoch (ours)	69.34%	56.88%	79.20%	85.32%
Qwen2VL-2B-80k-3epoch (ours)	70.54%	**62.92%**	80.43%	88.75%
Qwen2VL-7B-1M-3epoch (ours)	**85.2**%	-	**84.5%**	94%

Table 6: Results using Qwen2VL as base model

text, 0.578 for equations, 0.281 for tables) considering its compact size, especially when compared to similarly-sized models.

These results demonstrate that our lightweight model achieves a remarkable balance between model efficiency and OCR performance. The competitive or superior performance across multiple categories, particularly in English text and equation recognition, highlights the effectiveness of our training methodology and architectural choices. This efficiency-performance balance makes our solution particularly suitable for real-world applications with computational constraints, such as mobile devices or edge computing environments, without significantly sacrificing recognition quality compared to much larger models.

4.4 VQA Model

According to Table 6, the models trained on the Qwen2VL-2B base model show lower performance across all benchmarks compared to the baseline Qwen2VL-2B model. For example, the ChartQA test accuracy decreased from 71.48% for the baseline to 59.84%, 69.34%, and 70.54% for the different variants. Similarly, the OCRVQA test accuracy saw a more significant drop, from 74.65% to 43.37%, 56.88%, and 62.92%. Declines were also observed in the TextVQA validation and DocVQA test accuracies.

However, the Qwen2VL-7B model demonstrates improved performance on specific tasks. The ChartQA test accuracy for the Qwen2VL-7B-1M-3epoch model reached 85.2%, surpassing the baseline Qwen2VL-7B model's 83.0%. Additionally, in the TextVQA validation set, the accuracy improved slightly from the baseline 84.3% to 84.5%. This indicates that targeted parameter tuning and dataset optimization can enhance performance for particular tasks.

Encoder	Decoder	Param	Text		Equation		Table		Grounding
			EN	CN	EN	CN	EN	CN	F1-Score
SAM-ViT	OPT	94M+150M	0.265	0.293	0.566	0.720	0.719	0.714	-
SAM-ViT	Qwen1.5	94M+0.5B	0.252	0.251	0.468	0.572	0.591	0.587	0.682
SigLip	Qwen1.5	400M+0.5B	0.097	0.241	0.280	0.576	0.322	0.348	0.758
SigLip	Qwen2.5	400M+0.5B	0.081	0.214	0.277	0.578	0.283	0.281	0.764

Table 7: Ablation Study of Different Encoder and Decoder Combinations.

Overall, these findings highlight the significance of data quality and the specificity of datasets to the target application. While downsizing the model and reducing data quantity generally led to performance drops, there is potential to boost performance through customized dataset curation and benchmarking for specific use cases.

4.5 Ablation Study

Model Architecture. We conducted an ablation study to evaluate the impact of different encoder-decoder combinations on OCR performance, as shown in Table 7. Our findings reveal that the choice of visual encoder has a more significant impact on OCR performance than the choice of language decoder. Specifically, replacing the SAM-ViT encoder with SigLip resulted in substantial improvements across all metrics, with notable reductions in edit distance scores: from 0.252 to 0.097 for English text and from 0.591 to 0.322 for English tables.

The SigLip encoder, despite being larger (400M parameters) compared to SAM-ViT (94M parameters), provides superior feature extraction capabilities that significantly enhance text detection and recognition performance. Although SigLip is substantially larger than the SAM-ViT encoder, the inference time impact is negligible in practice. This is because during inference, the prefilling stage consumes considerably less time than the decoding stage for autoregressive generation. Therefore, the increased size of the encoder has minimal effect on overall inference speed while providing substantial accuracy benefits. The language decoder choice also influences performance, though to a lesser extent. Upgrading from Qwen1.5 to Qwen2.5 (both at 0.5B parameters) yielded modest but consistent improvements across most metrics, particularly for complex content types like tables. This suggests that even when parameter count remains constant, architectural improvements in newer model versions can translate to meaningful performance gains.

Model	Operation	Param	Text		Equation		Table		Grounding
			EN	CN	EN	CN	EN	CN	F1-Score
SigLip-Qwen	`spatial_unpad`	400M+0.5B	0.084	0.211	0.280	0.576	0.302	0.291	0.618
SigLip-Qwen	`resize`	400M+0.5B	0.081	0.214	0.277	0.578	0.283	0.281	0.764

Table 8: Ablation Study of Replacing Unpad with Resize.

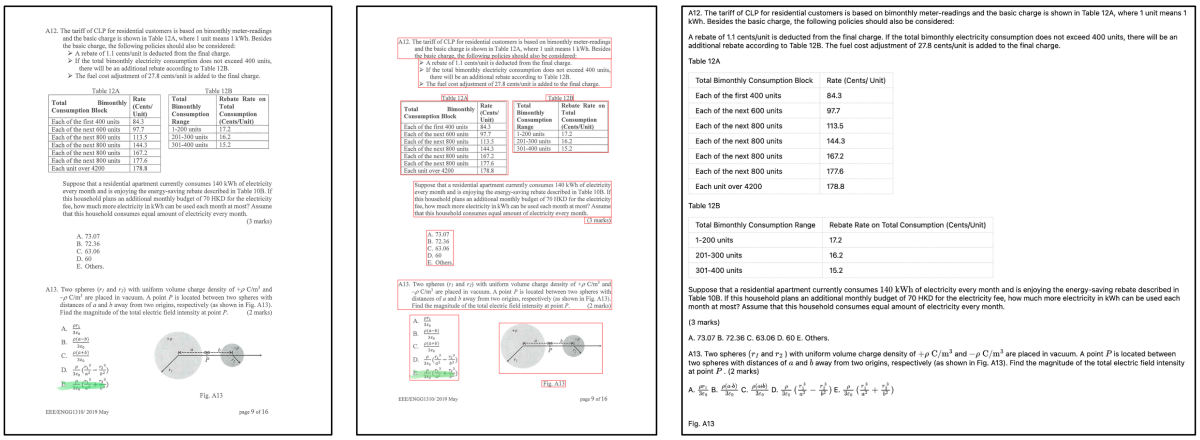
These findings demonstrate the importance of thoughtful model component selection, particularly highlighting how a more powerful visual encoder can significantly enhance overall system performance without substantially impacting inference latency. This efficiency makes our model particularly suitable for real-time applications where both accuracy and speed are critical considerations.

Removal of Spatial Unpad. As mentioned in section 3.2.2, we find that spatial unpad as the default operation used in Llava-Next after the AnyRes Matching step will cause performance degradation for grounding tasks. The reason of this is that the unpad operation are not compatable with the normalized coordinate we used for our training data, which will cause the model to learn the wrong coordinate system. We conducted an ablation study to evaluate the impact of replacing spatial unpad with resize on OCR performance, as shown in Table 8. Our findings reveal that the choice of operation has a significant impact on box F1 score, with minor improvement in OCR results probably due to the increase in visual tokens after removing unpad.

4.6 Qualitative Results

In this section, we present qualitative results that visually demonstrate our OCR system's capabilities. Despite the compact size of our model (only 0.5B parameters), it exhibits remarkable performance in document parsing tasks. Figure 12 showcases several examples of our model's text detection and recognition capabilities across diverse document types.

As evidenced by these examples, our lightweight model achieves high-quality results that are competitive with much larger models. The bounding box precision is particularly noteworthy, with accurate delineation of text regions even in documents with complex layouts and multiple columns. These results demonstrate that through careful optimization and training strategies, even compact models can achieve impressive OCR performance



Input Image ***Grounding Result*** ***Parsing Result***

Figure 12: Visualization of Document Parsing Results.

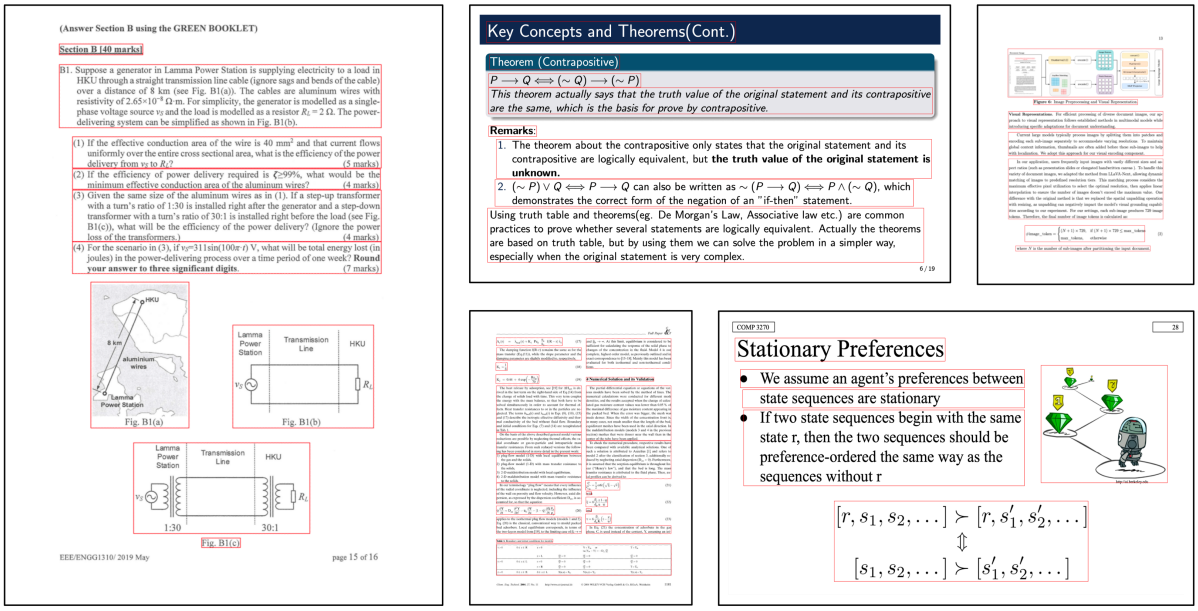


Figure 13: More Grounding Results.

that approaches that of much larger models while maintaining significantly lower computational requirements. This efficiency makes our solution particularly suitable for deployment in resource-constrained environments like mobile devices and edge computing scenarios.

4.7 Backend Implementation Details

Backend Architecture. The backend implementation adopts a Flask-based microservices architecture, a deliberate choice driven by Flask's lightweight nature and exceptional flexibility in handling asynchronous operations. This architectural approach enables the system to efficiently process multiple concurrent requests while maintaining a clean separation of concerns across different functional domains. Flask's minimal footprint and non-opinionated design philosophy allowed for custom tailoring of the application structure to match the specific requirements of this note-taking application, which demands both real-time interactivity and sophisticated file processing capabilities.

The system architecture is organized into three primary modules, each designed with clear boundaries and responsibilities. The API module houses all route handlers and endpoint definitions, implementing RESTful principles to provide a consistent and intuitive interface for client applications. This separation ensures that HTTP-specific logic remains properly isolated from core business rules, significantly enhancing maintainability as the system grows. The models module defines the application's domain objects with rigorous type specifications, establishing a clear contract for data structures throughout the application and preventing data integrity issues. Meanwhile, the services module encapsulates the core business logic and external integrations, particularly the Vision Language Model service, which serves as the foundation for the system's document analysis capabilities.

Flask Blueprints form the backbone of the routing system, providing a sophisticated mechanism for organizing endpoints into logical, domain-specific groups. The blueprint implementation demonstrates meticulous attention to separation of concerns:

- `files_bp` implements a comprehensive file handling system that supports:
  - Multiple file format processing (PDF, images, markdown)
  - Automatic content conversion
  - Unique file identification using UUID

- `canvas_bp` manages interactive canvas operations with:
  - Real-time canvas state management
  - Efficient JSON-based storage
  - Preview generation capabilities

This structured approach to route organization enables clean code modularization while preserving the semantic relationships between related endpoints. The files blueprint handles a comprehensive range of file operations, from initial upload and validation to format detection and content conversion. This component implements intelligent processing pipelines that can transform various input formats (PDFs, images, text documents) into standardized markdown representations using advanced AI capabilities. The canvas blueprint, meanwhile, orchestrates all interactions with the application's drawing capabilities, managing real-time canvas state operations through efficient JSON-based storage mechanisms. This approach allows for seamless canvas creation, retrieval, and modification while maintaining a persistent record of all changes.

The application's file storage system exemplifies a thoughtful approach to content organization, with directories structured according to content type and processing stage. The root uploads directory branches into specialized subdirectories: canvas for storing JSON representations of drawing data and their associated preview images; files for maintaining original, unmodified uploads in their native formats; and markdown for housing the processed, text-based representations of file content. This organization provides numerous advantages in terms of operational management, making backup procedures straightforward and enabling efficient content retrieval through predictable path structures. Furthermore, the clear separation of content types ensures that new formats can be incorporated without disrupting the existing architecture.

The overall architecture demonstrates a commitment to modern software engineering principles, particularly the separation of concerns and modularity. By compartmentalizing functionality into discrete, loosely-coupled components, the system achieves a high degree of maintainability and extensibility. New features can be added with minimal impact on existing code, and individual components can be refined or replaced as requirements evolve. This architectural approach also facilitates collaborative development, as different

team members can work on separate modules with clearly defined interfaces. The deliberate emphasis on modularity represents a forward-thinking design decision that anticipates future growth and adaptation of the application.

Vision Language Model Integration. The integration of Vision Language Models represents one of the most sophisticated aspects of the backend implementation, demonstrating a thoughtful approach to incorporating cutting-edge AI capabilities while maintaining system flexibility and resilience. The VLMService class serves as the cornerstone of this integration, implementing a provider-agnostic interface that abstracts away the complexities and idiosyncrasies of different AI service providers. This abstraction layer enables the application to leverage multiple VLM implementations—currently OpenAI's GPT-4 Vision and a custom SGLang implementation—while presenting a consistent, unified API to the rest of the system. The service's initialization clearly demonstrates this flexibility:

```
class VLMService:
    def __init__(
        self,
        provider: str = "openai", # "openai" or "sglang" for Different API format
        api_key: Optional[str] = None,
        model: str = "gpt-4o-2024-08-06",
        server_url: Optional[str] = None, # required for sglang
    ):
```

Listing 1: VLMService Constructor: Provider-agnostic AI service interface

This constructor design anticipates the rapid evolution of the AI landscape, positioning the application to easily adapt to new models and providers as they emerge without requiring substantial architectural changes. The parameterized initialization allows for runtime configuration of the service, enabling deployment across different environments with varying AI service availability. The default provider selection of "sglang" indicates a preference for custom, potentially self-hosted models, while still supporting commercial alternatives through the provider parameter.

The VLMService's implementation reflects careful consideration of both functional and non-functional requirements. Functionally, it provides two primary capabilities:

converting images to markdown representations and answering questions based on provided context files. The image-to-markdown functionality powers the system's document analysis capabilities, allowing it to extract structured content from various visual inputs such as scanned documents, handwritten notes, diagrams, and photographs. This capability transforms the application from a simple note repository to an intelligent content processing system that can make sense of diverse information sources. The question-answering functionality, meanwhile, implements a Retrieval Augmented Generation (RAG) approach, combining the power of large language models with context-specific information retrieval to provide accurate, relevant responses to user queries about their stored content.

From a non-functional perspective, the service implementation demonstrates attention to several critical quality attributes. Performance concerns are addressed through the use of asynchronous programming patterns, with both primary methods implemented as coroutines using Python's async/await syntax. This approach prevents long-running AI operations from blocking the main application thread, allowing the system to remain responsive even when processing complex documents or handling multiple concurrent requests. Resource management is carefully handled with temporary file cleanup and proper session handling in HTTP requests, preventing resource leaks that could degrade system performance over time. Configuration flexibility is achieved through environment variables that control provider selection, API keys, and endpoint URLs, enabling seamless deployment across different environments without code changes.

The implementation details reveal sophisticated error handling strategies that enhance system resilience. When interacting with external AI services, the code anticipates various failure modes—from network timeouts to API rate limits—and implements appropriate recovery mechanisms. The provider abstraction also enables failover capabilities; if one service becomes unavailable, the system could be configured to automatically switch to an alternative provider with minimal disruption. This focus on reliability is crucial for an application where AI capabilities are central to the user experience rather than merely supplementary features.

The choice to support multiple providers rather than relying exclusively on a single service reflects a strategic engineering decision with several advantages. It mitigates vendor lock-in risks, giving the application flexibility to adapt as the competitive landscape

and pricing models of AI services evolve. It provides opportunities for cost optimization by selecting different providers based on specific task requirements or budget constraints. It also enables the application to leverage specialized capabilities of different models; for instance, some vision models might excel at document analysis while others perform better with handwritten content. This heterogeneous approach ultimately delivers a more robust, adaptable system that can evolve alongside the rapidly changing AI technology landscape.

Data Model Design. The data model architecture represents a foundational aspect of the backend implementation, embodying domain-driven design principles that ensure both conceptual integrity and practical efficacy. Rather than adopting a simplistic approach based solely on database schema considerations, the design process began with a thorough analysis of the domain concepts that underpin the application's functionality. This approach resulted in a rich object model that accurately captures the semantics of notes, canvases, and documents, providing a robust foundation for all application features while maintaining a clear separation between different logical entities.

The Canvas model exemplifies this domain-driven approach, implementing a sophisticated composite pattern to represent the complex structure of drawing spaces:

```
class Canvas:
    id: str
    title: str
    width: int
    height: int
    background_color: str
    elements: List[CanvasElement]
    created_at: datetime
    updated_at: datetime
```

Listing 2: Canvas Model: Hierarchical representation of interactive drawing space

This class structure succinctly captures both the intrinsic properties of a canvas (dimensions, background color) and its compositional nature as a container for multiple drawing elements. Each Canvas instance serves as a container for multiple CanvasElement objects, which themselves can represent diverse content types such as strokes, text

```json
{
  "id": "canvas-123",
  "title": "My Canvas",
  "width": 800,
  "height": 600,
  "background_color": "#FFFFFF",
  "elements": [
    {
      "id": "stroke-1",
      "type": "stroke",
      "data": {
        "points": [{"x": 100, "y": 100, "pressure": 0.5, "tilt": 0}],
        "color": "#000000",
        "brush_size": 2.0,
        "brush_type": "pen"
      },
      "created_at": "2024-04-06T12:00:00",
      "updated_at": "2024-04-06T12:00:00"
    },
    {
      "id": "text-1",
      "type": "text",
      "data": {
        "content": "Hello World",
        "position": {"x": 200, "y": 200},
        "font_family": "Arial",
        "font_size": 16,
        "color": "#000000",
        "style": {"bold": true, "italic": false, "underline": false}
      },
      "created_at": "2024-04-06T12:01:00",
      "updated_at": "2024-04-06T12:01:00"
    }
  ],
  "created_at": "2024-04-06T12:00:00",
  "updated_at": "2024-04-06T12:01:00"
}
```


Figure 14: JSON representation of a Canvas object, illustrating the serialization of properties and elements into a structured format.

annotations, or embedded images. This hierarchical organization mirrors the natural conceptual model of digital drawings, where a canvas holds various elements with different properties and behaviors. The implementation carefully tracks both structural attributes and temporal metadata (creation and modification timestamps), enabling not only accurate rendering but also effective versioning and history tracking. This comprehensive model supports complex operations such as element manipulation, canvas resizing, and state restoration, all while maintaining coherent semantic boundaries that reflect user mental models. Figure 14 illustrates the JSON representation of a Canvas object, showcasing the serialization of its properties and elements into a structured format suitable for storage and transmission.

The Document model demonstrates a different but equally sophisticated design approach, focusing on the representation and lifecycle management of uploaded files:

```
class Document:
    id: str
    title: str
    content_type: str
    file_path: str
    markdown_path: Optional[str]
    metadata: Dict[str, Any]
```

Listing 3: Document Model: File management with support for derived representations

This model serves as a central entity for file operations, bridging the gap between raw file storage and semantic content interpretation. The implementation maintains a clear distinction between the original uploaded content (preserved at its original `file_path`) and any derived representations (such as markdown conversions at `markdown_path`), facilitating both content fidelity and processing flexibility. The inclusion of a flexible metadata field, implemented as a dictionary, provides an extensible mechanism for capturing file-specific attributes without requiring model modifications for each new attribute type. This design decision anticipates the diverse range of file formats and associated properties that users might upload, ensuring the system can accommodate this variety without sacrificing type safety or descriptive accuracy.

The processing pipeline for file handling represents a particularly sophisticated aspect of the data model implementation. Upon upload, files undergo a structured sequence of operations: validation based on MIME type and extension, storage with unique identifiers, content analysis using AI services (when appropriate), and metadata extraction. This linear process ensures consistent handling across diverse file types while maintaining clear boundaries between processing stages. For PDF documents, the system implements an especially advanced processing flow, converting each page to an image at an appropriate resolution, then leveraging the VLM service to extract structured content, and finally assembling these page-level extractions into a coherent markdown document. This approach effectively bridges the gap between unstructured visual content and structured, searchable text, significantly enhancing the utility of uploaded documents.

The decision to use strongly typed classes rather than simple dictionaries or database-mapped objects reflects a commitment to type safety and domain clarity. By defining explicit class structures with well-defined properties and methods, the implementation prevents a wide range of potential errors while providing clear interfaces for other system components. This approach also enables comprehensive validation at the object level, ensuring data integrity throughout the application lifecycle. The `to_dict` and `from_dict` methods implemented for each model facilitate seamless serialization and deserialization, supporting both persistent storage and API interactions without compromising the rich semantic structure of the domain objects. This bidirectional transformation capability ensures that the models can participate in both internal processing logic and external data exchange without requiring duplicate implementations or compromising conceptual integrity.

The overall data model design demonstrates thoughtful consideration of both immediate functional requirements and long-term evolutionary needs. By establishing clear domain boundaries, implementing flexible extension points, and maintaining rigorous type safety, the architecture provides a solid foundation that can accommodate new features and refinements without requiring disruptive structural changes. This forward-looking approach represents a significant engineering investment that will yield dividends throughout the application lifecycle, reducing maintenance costs and enabling rapid feature development while preserving conceptual clarity and data integrity.

4.8 Desktop Frontend Development

The development of OmniNote has made substantial strides, resulting in a robust application that demonstrates the core functionalities outlined in the key feature set.

Canvas-Based Note-Taking. The central canvas interface is fully operational, enabling users to create and update notes using a combination of text entry, freehand drawing tools, and inserted images. This flexible workspace forms the foundation for rich, multimodal note-taking experiences.

Hierarchical File Management. Users can organize their content through a structured folder system. Users can create several workspaces. In each workspace, they can create a canvas together with several uploaded files. Our app supports all kinds of file formats.

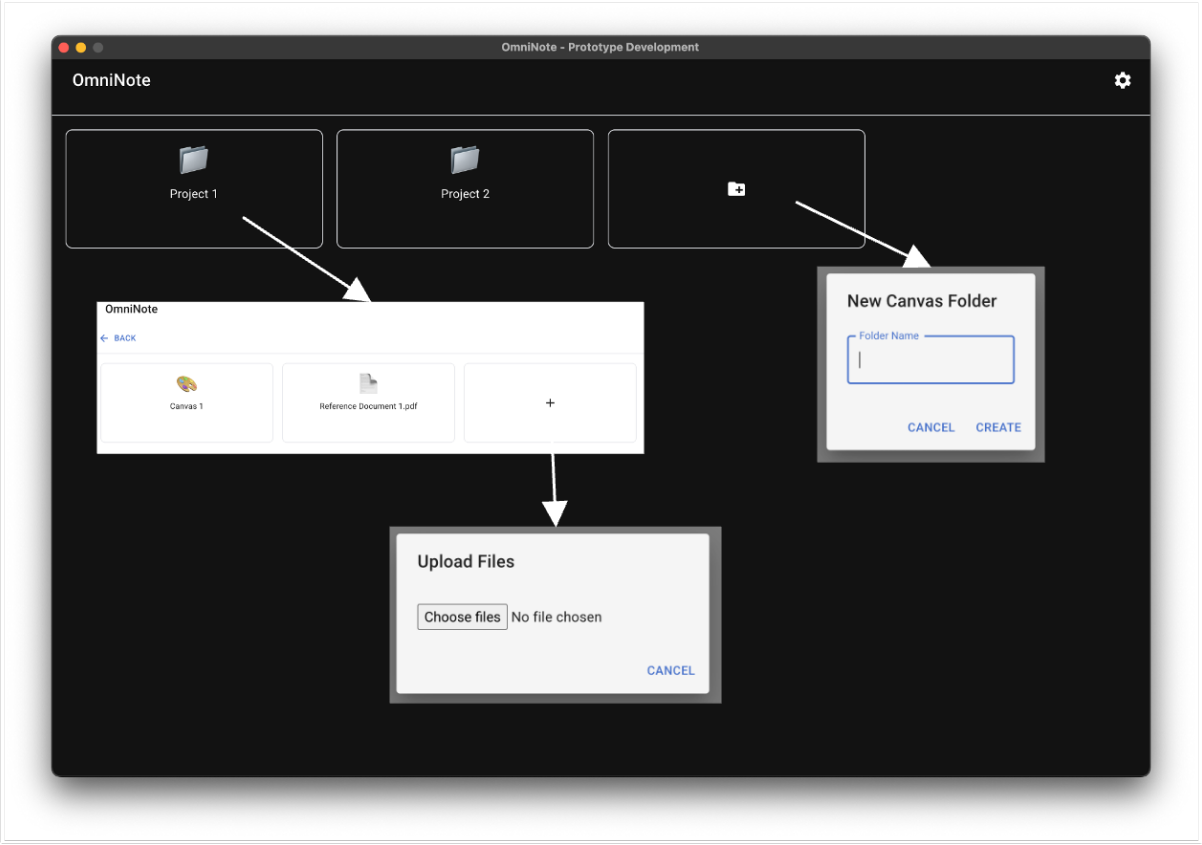


Figure 15: Design and functionality of the note management system in desktop application

Additionally, the canvas and all files will be easily converted to the markdown format through our model in the server side. Uploaded files are seamlessly integrated into the application.

Interactive Model Integration. The application connects with backend OCR (Optical Character Recognition) and VQA (Visual Question Answering) APIs, enabling users to convert canvas into markdown formats, extract text and ask context-based questions about their notes, diagrams, and related files in the workspace. The QA interface is interactive and supports model selection, image embedding, and file attachment.

Responsive and Customizable Interface. : The user interface has been designed with responsiveness in mind, adapting fluidly to various screen sizes and platforms. Both light

and dark mode options are available, catering to user comfort and accessibility across usage contexts.

An overview of the desktop application's design and features is illustrated in **??** and Figure 15, showcasing functionalities such as model selection, canvas interactions, file handling, and UI customization. Together, these elements deliver a cohesive and intelligent platform tailored to modern note management needs.

4.9 Mobile Application Frontend Development

To enhance the convenience of handwriting and diagram sketching, a dedicated mobile application has been developed alongside the desktop version. This mobile app focuses on providing an optimized experience for touch-based input and free-form handwriting, making it especially effective for handwritten note-taking and on-the-go ideation.

The mobile frontend was implemented using React Native in conjunction with Expo Go, allowing for rapid development, cross-platform compatibility, and seamless testing across iOS and Android devices. The architecture follows component-based design principles, ensuring maintainability and scalability as the application evolves.

Particular attention was paid to preserving a consistent user experience across platforms. The mobile interface replicates the visual and functional structure and features of the desktop version, including layout, navigation flow, and theming (light/dark modes). Shared design tokens and reusable UI components were employed to unify the appearance across devices, while responsive touch interactions were introduced to accommodate mobile-specific behaviors. Detailed interface designs are presented in the appendix.

4.10 Case Studies

To demonstrate OmniNote in realistic use scenarios, we present two case studies that illustrate its capabilities in both document understanding and multimodal interaction.

Case 1: Parsing a Handwritten Math Note. The user uploads a scanned handwritten page containing math formulas, bullet points, and a diagram. The parsing model first perform document layout analysis on the document page then performs text recognition, extracting each block with bounding boxes. Equations are rendered in LaTeX and displayed above each line. The user then asks: *"What does the second equation represent?"*

The system routes this to the Qwen2-VL (QA) model, which returns: *"It represents the derivative of the loss function with respect to weight W."* The visual context and chat history are preserved for continued interaction.

Case 2: Understanding a Chart in a Scientific PDF. A multi-page research paper PDF is uploaded. On page 5, a bar chart compares accuracy across models. The user quotes the chart region and asks: *"Which model achieved the highest accuracy?"* OmniNote extracts the chart using its chart parser and identifies axis labels and bars. The grounded visual QA module responds: *"Model D achieved the highest accuracy at 94.2%."* The response is answered in the chat panel.

These case studies demonstrate OmniNote's seamless integration of document parsing, visual understanding, and natural-language interaction in real academic workflows.

5 Ethical Considerations and Privacy

While OmniNote focuses on the technical challenge of multimodal document understanding, ethical and privacy-related considerations are equally critical to ensure the system's safe and responsible deployment. This chapter explores how OmniNote will address these aspects, and highlights areas that require ongoing attention.

5.1 User Data Privacy

OmniNote processes user-provided content such as handwritten notes, scanned textbooks, and personal annotations. To protect user privacy, the system is designed to minimize persistent storage of user data. All uploaded files can be stored in a temporary AWS S3 bucket with restricted access. Files are automatically deleted after inference is complete. This ensures that the backend only uses the data for real-time inference and never retains identifiable content.

Additionally, all API requests can be authenticated using JWT (JSON Web Tokens). This prevents unauthorised access and ensures session-level integrity. Frontend clients and backend servers communicate exclusively over HTTPS, ensuring transport layer encryption.

5.2 Bias in Model Predictions

The fine-tuned models, such as GOT and Qwen2-VL, can be based on large pre-trained foundations that may inherit biases present in their training data. This includes biases related to language, content region, and even handwriting styles. For example, if most handwriting training data consists of neatly scanned English notes, the model may underperform on less common scripts, informal layouts, or multilingual documents.

To mitigate this, the OmniNote-1M dataset incorporates both printed and handwritten samples in English and Chinese. However, more work is needed to expand representation, especially for low-resource scripts and culturally diverse document types. Future iterations may explore adversarial training or debiasing techniques [10].

5.3 Responsible Use and Transparency

As AI-assisted note-taking tools become more widely adopted in education and workplace environments, it is essential to ensure users understand their limitations. OmniNote does not store or transmit content outside the designated infrastructure and cannot make inferences beyond the input document. It also does not learn from user interactions unless explicitly authorised.

To enhance transparency, users are shown a visual overlay of model-predicted regions, enabling them to verify the accuracy of parsing and answer grounding. Where confidence scores are available (e.g., from token probabilities), these can be surfaced to guide user trust [41].

5.4 Predictive Privacy

The concept of predictive privacy addresses the ethical concerns arising when AI systems infer sensitive information about individuals without their explicit consent. This includes scenarios where models predict personal attributes or behaviors based on available data, potentially leading to privacy violations even if the inferred information is not directly disclosed [31].

6 Conclusion

In conclusion, this report presents the development of our AI-powered note-taking application that utilizes vision-language models to improve productivity and user experience in note-taking scenarios. The primary objectives were to deliver a note-taking app capable of transforming drafts into organized formats, generating notes from lecture materials, and providing advanced QA features using VLMs. To achieve these goals, we conducted a thorough literature review of recent advances in VLMs for document analysis task, developed a system architecture, selected baseline models, and started fine-tuning for our use case. Additionally, we collected a comprehensive dataset to ensure the system's applicability for note-taking purposes.

The major findings of the investigation indicate that preliminary results from the fine-tuned models show promising performance in OCR and DocQA tasks, particularly when adapted to the note-taking context. Early-stage experiment results suggest that our fine-tuned 0.5B model outperforms baseline models across multiple tasks, including visual grounding, text retrieval, and multimedia content understanding. This highlights the effectiveness of our self-collected dataset in providing diverse and high-quality training data tailored to the specific requirements of note-taking tasks. The investigation's contribution lies in demonstrating the potential of integrating vision-language models for efficient note-taking solutions. By combining OCR with DocQA tasks with vison-language models, this project offers an approach that can be further refined for broader use cases, potentially enhancing the user experience in both academic and professional settings.

Limitations During model deployment we faced challenges of running AI models on mobile devices, particularly lower-end phones and tablets , we opted to deploy our system using SGLang instead of implementing on-device inference. This cloud-based deployment strategy allows us to leverage more powerful computational resources while maintaining accessibility across a range of devices. The dataset, while specifically tailored for note-taking scenarios, may limit the system's ability to handle a wider variety of document types. Future work should focus on expanding the dataset to cover more diverse document types and optimizing the deployment architecture to balance server costs with user experience. Testing the system in real-world environments will also be crucial to

gather concrete feedback for further optimizations, ensuring the system can better meet the demands of a broader range of use cases.

Future Works Future work will focus on quantization and structured pruning techniques to ensure model deployability on edge devices. Additionally, we plan to enhance our dataset by generating high-quality synthetic note-taking data that encompasses a broader range of content types. Currently, our synthetic data primarily consists of plain text and handwritten mathematical formulas. However, real-world notes often include more varied elements such as handwritten tables, diagrams, and other graphical content. Advancements in handwriting synthesis techniques can facilitate the creation of such diverse synthetic data. For instance, models like the Handwritten Math Generator can produce synthetic handwritten mathematical expressions, aiding in training deep learning models.

By incorporating these methods, we can create a more comprehensive synthetic dataset that better reflects the complexity of real-world notes. This, in turn, would improve the robustness and applicability of our AI-powered note-taking application across various academic and professional contexts.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, H. Touvron, H. Jegou, I. Misra, B. Mildenhall, A. Zisserman, et al. "Flamingo: a Visual Language Model for Few-Shot Learning". In: *arXiv preprint arXiv:2204.14198* (2022).

[3] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, H. Zhong, Y. Zhu, M. Yang, Z. Li, J. Wan, P. Wang, W. Ding, Z. Fu, Y. Xu, J. Ye, X. Zhang, T. Xie, Z. Cheng, H. Zhang, Z. Yang, H. Xu, and J. Lin. *Qwen2.5-VL Technical Report*. 2025. arXiv: 2502.13923 [cs.CV]. URL: https://arxiv.org/abs/2502.13923.

[4] D. Baviskar, S. Ahirrao, K. Kotecha, et al. "ICDAR 2023 Competition on Visual Question Answering on Business Document Images". In: *17th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. 2023, pp. 1234–1241.

[5] A. F. Biten, R. Tito, A. Mafla, L. Gomez, M. Rusinol, D. Karatzas, C. V. Jawahar, and E. Valveny. "Scene Text Visual Question Answering". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4291–4300.

[6] L. Blecher, G. Cucurull, T. Scialom, and R. Stojnic. "Nougat: Neural optical understanding for academic documents". In: *arXiv preprint arXiv:2308.13418* (2023).

[7] Z. Cai, M. Cao, H. Chen, K. Chen, K. Chen, X. Chen, X. Chen, Z. Chen, Z. Chen, P. Chu, et al. "Internlm2 technical report". In: *arXiv preprint arXiv:2403.17297* (2024).

[8] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, et al. "Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 24185–24198.

[9] *Claude 3 model card*. https://docs.anthropic.com/en/docs/resources/claude-3-model-card.

[10] M. M. Danquah, P. S. Dadzie, K. Gyesi, F. Yeboah, and C. Y. Nyarko. "Artificial intelligence implementation strategies for Ghanaian academic libraries: A scoping review". In: *The Journal of Academic Librarianship* 50.6 (2024), p. 102975.

[11] Y. Ding, Y. Wang, Y. Wang, Y. Li, and Z. Zhao. "PDF-VQA: A New Dataset for Real-World VQA on PDF Documents". In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer. 2023, pp. 547–563.

[12] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. "The llama 3 herd of models". In: *arXiv preprint arXiv:2407.21783* (2024).

[13] L. Fu, B. Yang, Z. Kuang, J. Song, Y. Li, L. Zhu, Q. Luo, X. Wang, H. Lu, M. Huang, Z. Li, G. Tang, B. Shan, C. Lin, Q. Liu, B. Wu, H. Feng, H. Liu, C. Huang, J. Tang, W. Chen, L. Jin, Y. Liu, and X. Bai. *OCRBench v2: An Improved Benchmark for Evaluating Large Multimodal Models on Visual Text Localization and Reasoning*. 2024. arXiv: 2501.00321 [cs.CV]. URL: https://arxiv.org/abs/2501.00321.

[14] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. "Making the v in vqa matter: Elevating the role of image understanding in visual question answering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6904–6913.

[15] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks". In: *Proceedings of the 23rd International Conference on Machine Learning*. ACM. 2006, pp. 369–376.

[16] M. A. Hossain and S. Afrin. "Optical Character Recognition based on Template Matching". In: *Global Journal of Computer Science and Technology* 19.2 (2019).

[17] A. K. Jain, R. P. W. Duin, and J. Mao. "Statistical Pattern Recognition: A Review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.1 (2000), pp. 4–37.

[18] G. Jaume, H. K. Ekenel, and J.-P. Thiran. "FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents". In: *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*. IEEE. 2019, pp. 1–6.

[19] S. Kaushik. "Deep Learning Approaches for OCR". In: *LinkedIn* (2021). https://www.linkedin.com/pulse/deep-learning-approaches-ocr-shivansh-kaushik.

[20] G. Kim, T. Hong, M. Yim, J. Nam, J. Park, J. Yim, W. Hwang, S. Yun, D. Han, and S. Park. "OCR-Free Document Understanding Transformer". In: *European Conference on Computer Vision*. Springer. 2022, pp. 498–517.

[21] G. Kim, T. Hong, M. Yim, J. Nam, J. Park, J. Yim, W. Hwang, S. Yun, D. Han, and S. Park. "Ocr-free document understanding transformer". In: *European Conference on Computer Vision*. Springer. 2022, pp. 498–517.

[22] M. Li, Y. Zhang, L. Zhang, H. Li, L. Zhou, N. Li, S. Liu, Y. Wang, L. Hou, W. Chen, et al. "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2021, pp. 981–993.

[23] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang. "CASIA online and offline Chinese handwriting databases". In: *International Journal on Document Analysis and Recognition* 14.1 (2011), pp. 15–22.

[24] F. Liu, X. Wang, W. Yao, J. Chen, K. Song, S. Cho, Y. Yacoob, and D. Yu. "MMC: Advancing Multimodal Chart Understanding with Large-scale Instruction Tuning". In: *arXiv preprint arXiv:2311.10774* (2023).

[25] Y. Liu, L. Jin, S. Zhang, C. Luo, and S. Zhang. "OCRBench: On the Hidden Mystery of OCR in Large Multimodal Models". In: *Science China Information Sciences* 67.12 (2024). DOI: 10.1007/s11432-024-4235-6. URL: https://doi.org/10.1007/s11432-024-4235-6.

[26] Y. Liu, B. Yang, Q. Liu, Z. Li, Z. Ma, S. Zhang, and X. Bai. "Textmonkey: An ocr-free large multimodal model for understanding document". In: *arXiv preprint arXiv:2403.04473* (2024).

[27] U.-V. Marti and H. Bunke. "The IAM-database: an English sentence database for offline handwriting recognition". In: *International Journal on Document Analysis and Recognition* 5.1 (2002), pp. 39–46.

[28] M. Mathew, D. Karatzas, and C. V. Jawahar. "DocVQA: A Dataset for VQA on Document Images". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 2200–2209.

[29] J. Michael, R. Labahn, T. Grüning, and J. Zöllner. "Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition". In: *arXiv preprint arXiv:1903.07377* (2019).

[30] A. Mishra, S. Shekhar, A. K. Singh, and A. Chakraborty. "Ocr-vqa: Visual question answering by reading text in images". In: *2019 international conference on document analysis and recognition (ICDAR)*. IEEE. 2019, pp. 947–952.

[31] R. Mühlhoff. "Predictive privacy: towards an applied ethics of data analytics". In: *Ethics and Information Technology* 23.4 (2021), pp. 675–690.

[32] A. Nassar, A. Marafioti, M. Omenetti, M. Lysak, N. Livathinos, C. Auer, L. Morin, R. T. de Lima, Y. Kim, A. S. Gurbuz, et al. "SmolDocling: An ultra-compact vision-language model for end-to-end multi-modal document conversion". In: *arXiv preprint arXiv:2503.11576* (2025).

[33] OpenAI. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[34] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, and F. Wei. "Kosmos-2: Grounding multimodal large language models to the world". In: *arXiv preprint arXiv:2306.14824* (2023).

[35] T. Plötz and G. A. Fink. "Markov Models for Offline Handwriting Recognition: A Survey". In: *International Journal on Document Analysis and Recognition* 12.4 (2009), pp. 269–298.

[36] R. Powalski, Ł. Borchmann, D. Jurkiewicz, T. Dwojak, M. Pietruszka, and G. Pałka. "Going Full-TILT Boogie on Document Understanding with Text-Image-Layout Transformer". In: *International Conference on Document Analysis and Recognition*. Springer. 2021, pp. 732–747.

[37] J. Poznanski, J. Borchardt, J. Dunkelberger, R. Huff, D. Lin, A. Rangapur, C. Wilhelm, K. Lo, and L. Soldaini. "olmOCR: Unlocking Trillions of Tokens in PDFs with Vision Language Models". In: *arXiv preprint arXiv:2502.18443* (2025).

[38] A. Singh, V. Natarajan, X. Jiang, X. Chen, M. Rohrbach, D. Batra, and D. Parikh. "TextOCR: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8802–8812.

[39] A. Singh, V. Natarajan, X. Jiang, X. Chen, M. Rohrbach, D. Batra, and D. Parikh. "Towards VQA Models That Can Read". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8317–8326.

[40] R. Tanaka, K. Nishida, K. Nishida, T. Hasegawa, I. Saito, and K. Saito. "SlideVQA: A Dataset for Document Visual Question Answering on Multiple Images". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 11. 2023, pp. 13636–13645.

[41] S. Venkatasubbu and G. Krishnamoorthy. "Ethical considerations in AI addressing bias and fairness in machine learning models". In: *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)* 1.1 (2022), pp. 130–138.

[42] R. Verma. "A Survey of Feature Extraction and Classification Techniques in OCR Systems". In: *International Journal of Computer Applications & Information Technology* 1.3 (2012), pp. 1–5.

[43] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, et al. "Yolov10: Real-time end-to-end object detection". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 107984–108011.

[44] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, et al. "Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution". In: *arXiv preprint arXiv:2409.12191* (2024).

[45] H. Wei, L. Kong, J. Chen, L. Zhao, Z. Ge, J. Yang, J. Sun, C. Han, and X. Zhang. "Vary: Scaling up the vision vocabulary for large vision-language models". In: *arXiv preprint arXiv:2312.06109* (2023).

[46] H. Wei, C. Liu, J. Chen, J. Wang, L. Kong, Y. Xu, Z. Ge, L. Zhao, J. Sun, Y. Peng, et al. "General OCR Theory: Towards OCR-2.0 via a Unified End-to-end Model". In: *arXiv preprint arXiv:2409.01704* (2024).

[47] Y. Xu, T. Xu, T. Lv, L. Cui, F. Wei, G. Wang, Y. Lu, D. Florencio, C. Zhang, and L. Zhou. "LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding". In: *arXiv preprint arXiv:2012.14740* (2020).

[48] Z. Yang, J. Tang, Z. Li, P. Wang, J. Wan, H. Zhong, X. Liu, M. Yang, P. Wang, S. Bai, L. Jin, and J. Lin. *CC-OCR: A Comprehensive and Challenging OCR Benchmark for Evaluating Large Multimodal Models in Literacy*. 2024. arXiv: 2412.02210 [cs.CV]. URL: https://arxiv.org/abs/2412.02210.

[49] X. Ye, Y. Liu, Y. Wang, Y. Li, and Z. Zhao. "UReader: Universal OCR-free Visually-situated Language Understanding with Multimodal Large Language Model". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. 2023, pp. 2345–2356.

[50] Y. Yuan, X. Liu, W. Dikubab, H. Liu, Z. Ji, Z. Wu, and X. Bai. "Syntax-Aware Network for Handwritten Mathematical Expression Recognition". In: *arXiv preprint arXiv:2203.01601* (2022).

[51] Z. Zhao, Y. Li, Y. Zhang, Y. Wang, H. Zhang, Y. Li, Y. Liu, Y. Wang, Y. Li, and Z. Zhao. "ScreenQA: Large-Scale Question-Answer Pairs over Mobile App Screenshots". In: *arXiv preprint arXiv:2209.08199* (2022).

[52] A. Zou, W. Yu, H. Zhang, K. Ma, D. Cai, Z. Zhang, H. Zhao, and D. Yu. "DOCBENCH: A Benchmark for Evaluating LLM-based Document Reading Systems". In: *arXiv preprint arXiv:2407.10701* (2024).

Appendix

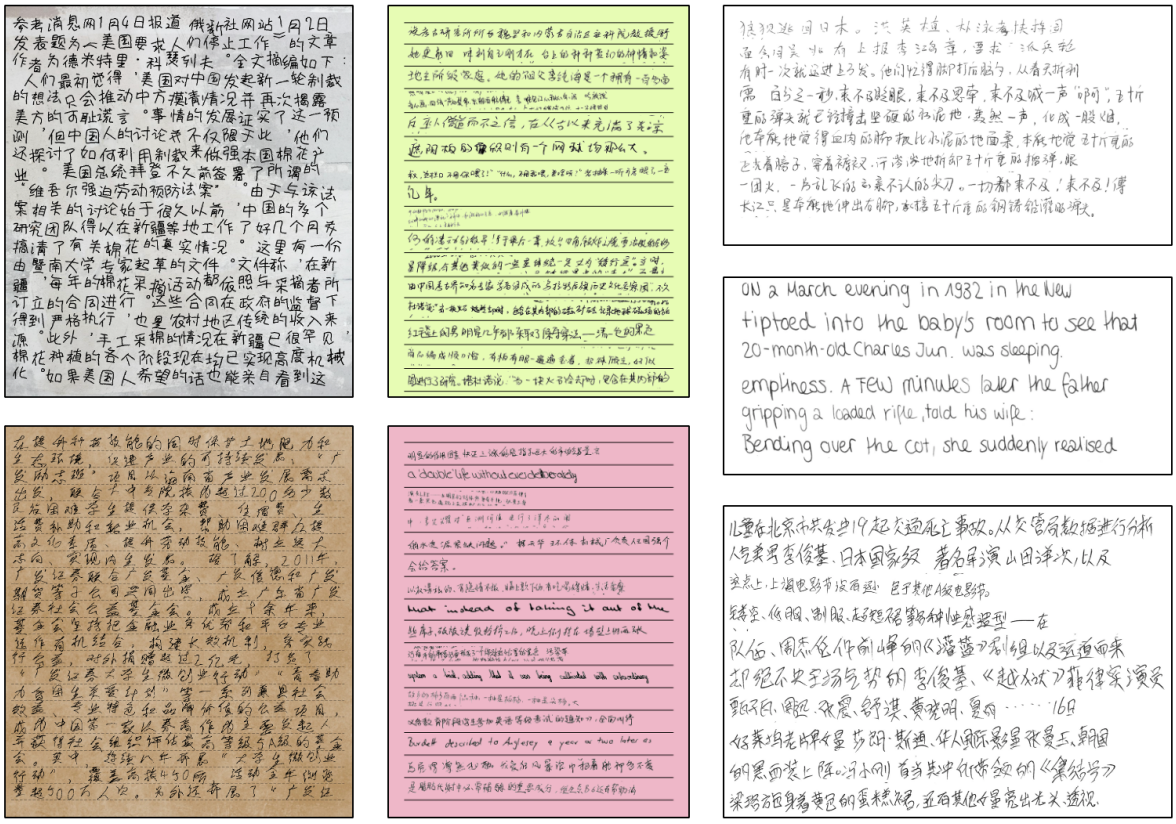


Figure 16: Examples of Synthetic Handwritten Text in Section 3.3.3.

Figure 17: Examples of Synthetic Printed Text in Section 3.3.3.

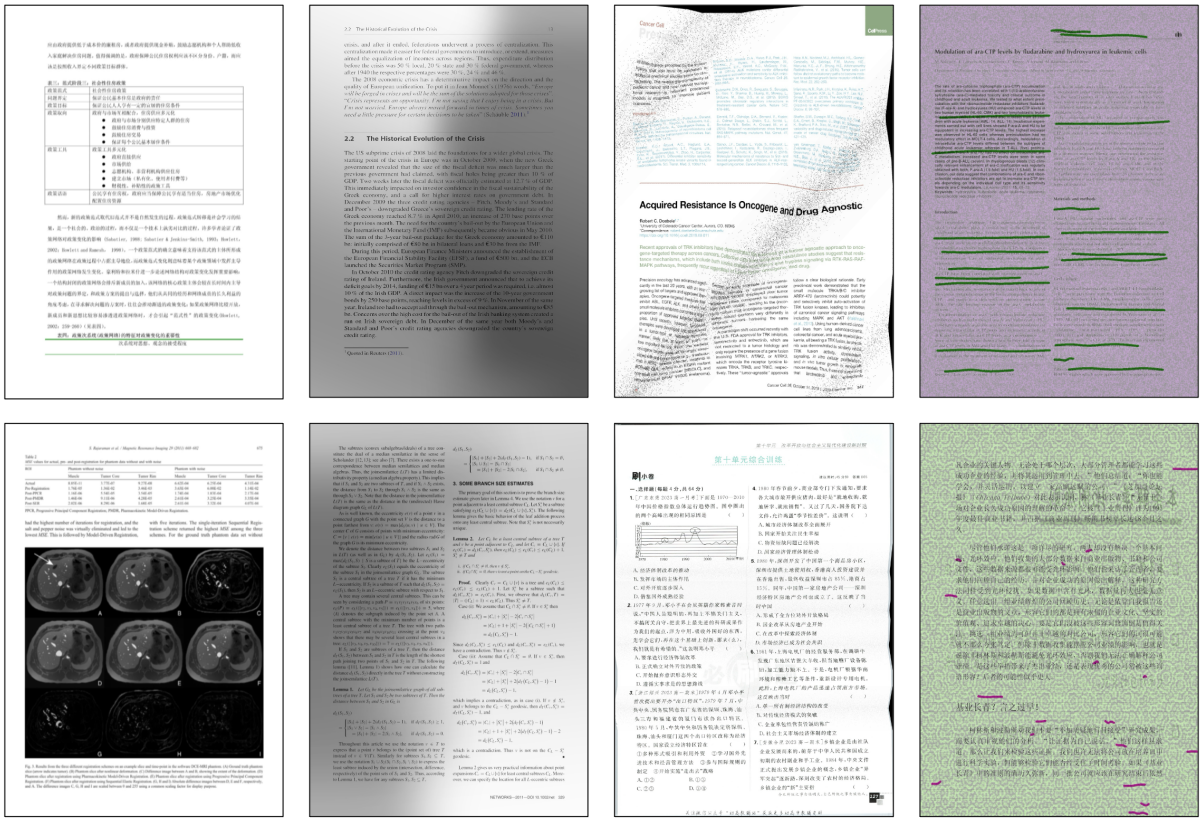


Figure 18: Examples of Image Augmentation mentioned in Section 3.3.6.

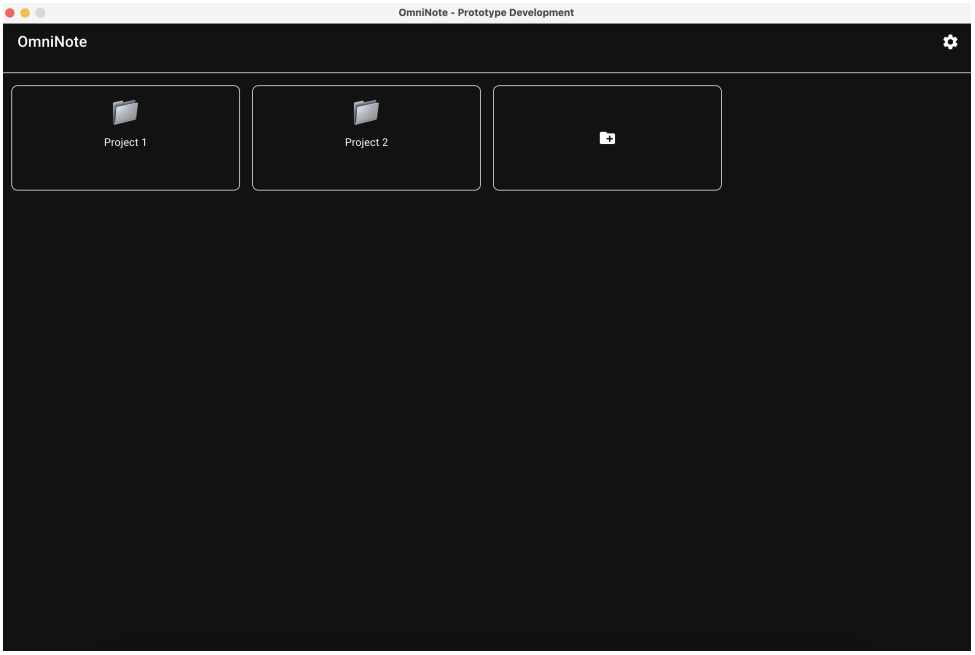


Figure 19: Desktop Application: Note Management System

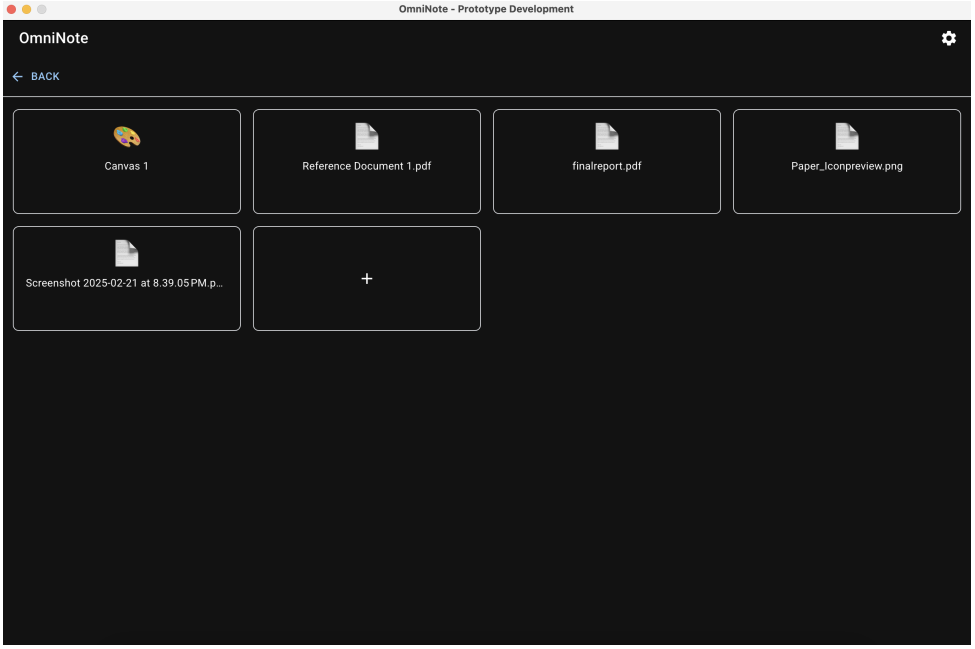


Figure 20: Desktop Application: Note Management System

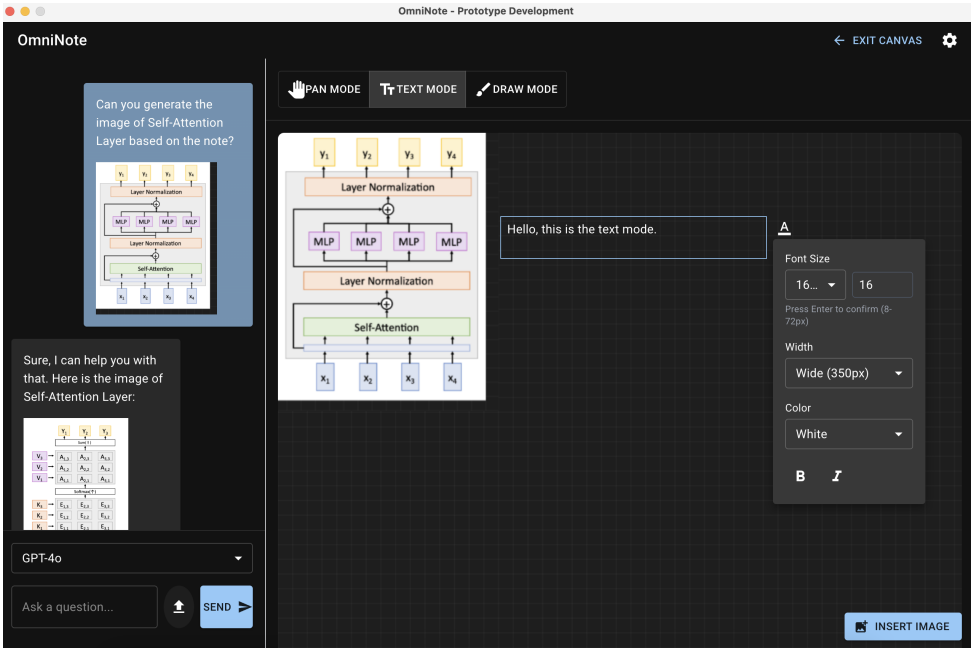


Figure 21: Desktop Application: Note Taking System

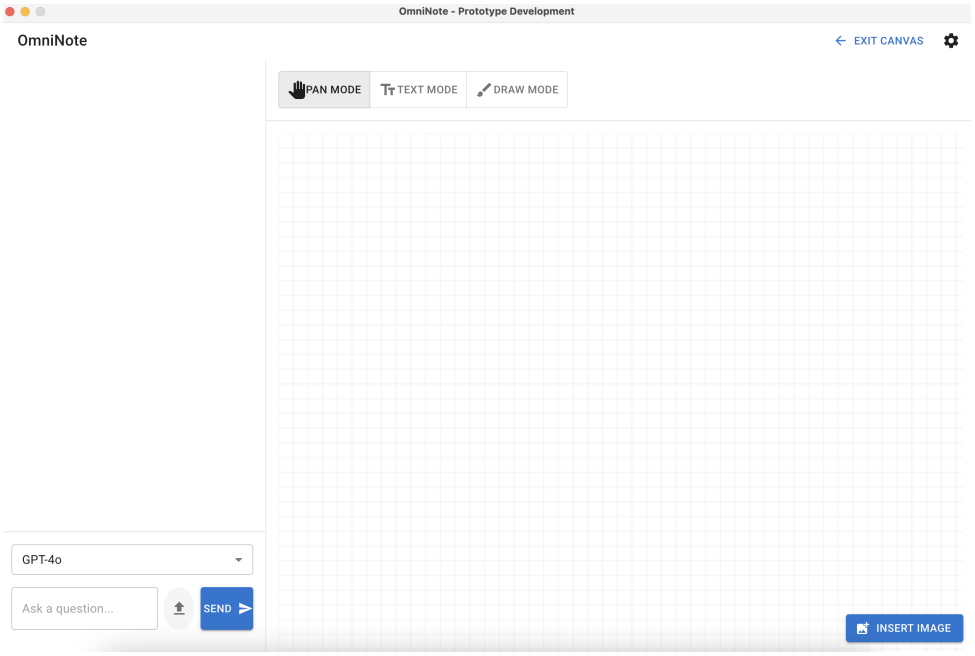


Figure 22: Desktop Application: Light Theme

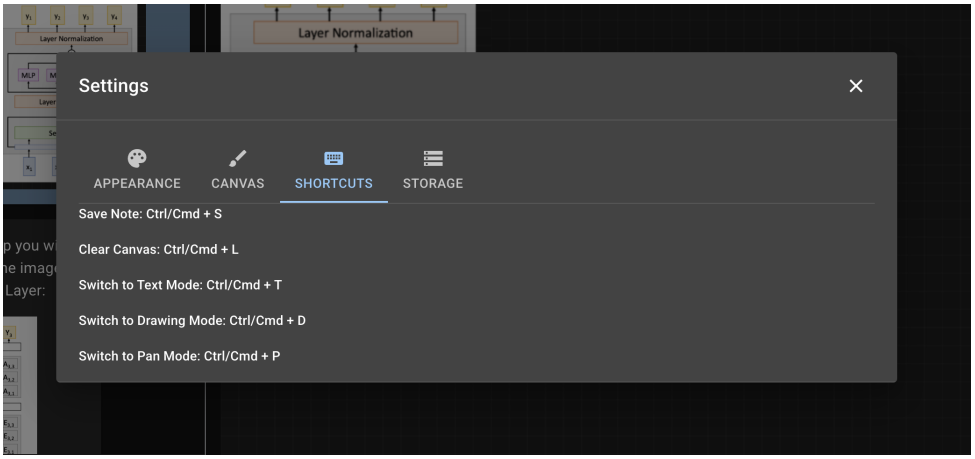


Figure 23: Desktop Application: Settings

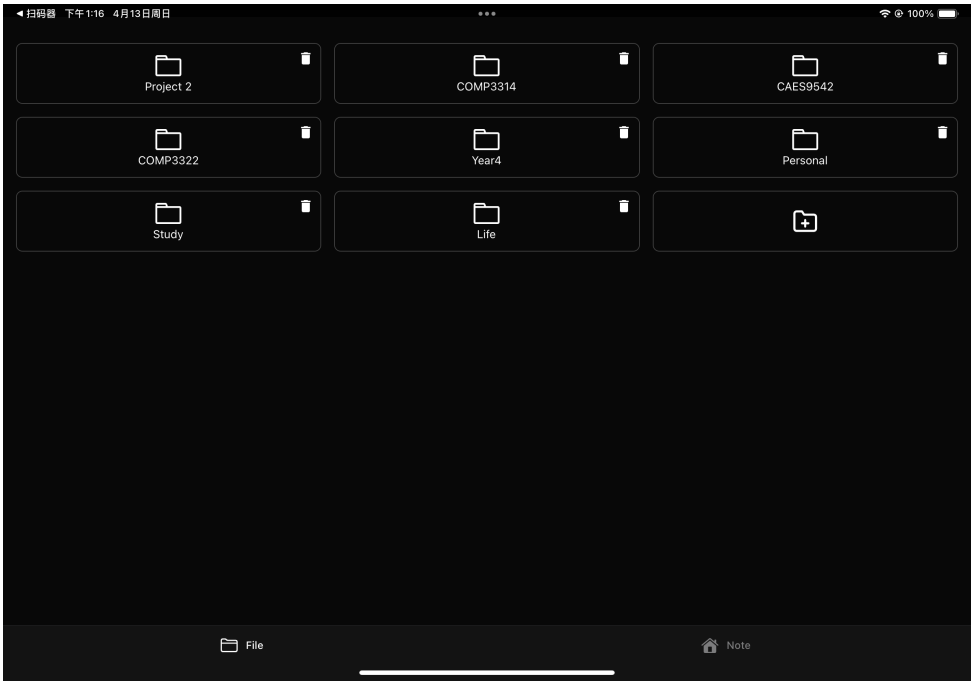


Figure 24: Mobile Application: Note Management System

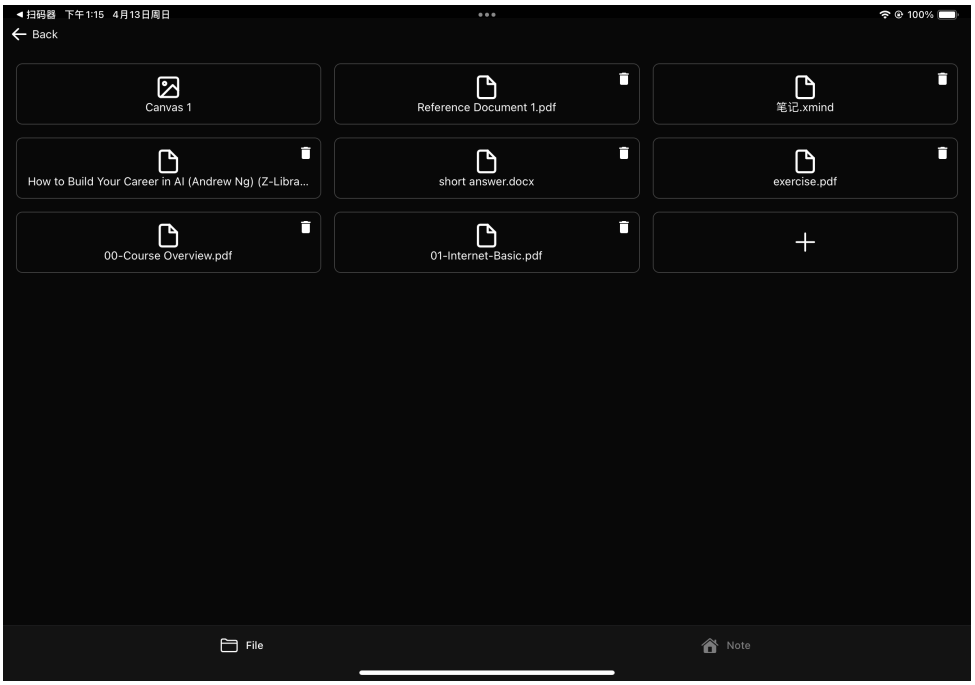


Figure 25: Mobile Application: Note Management System

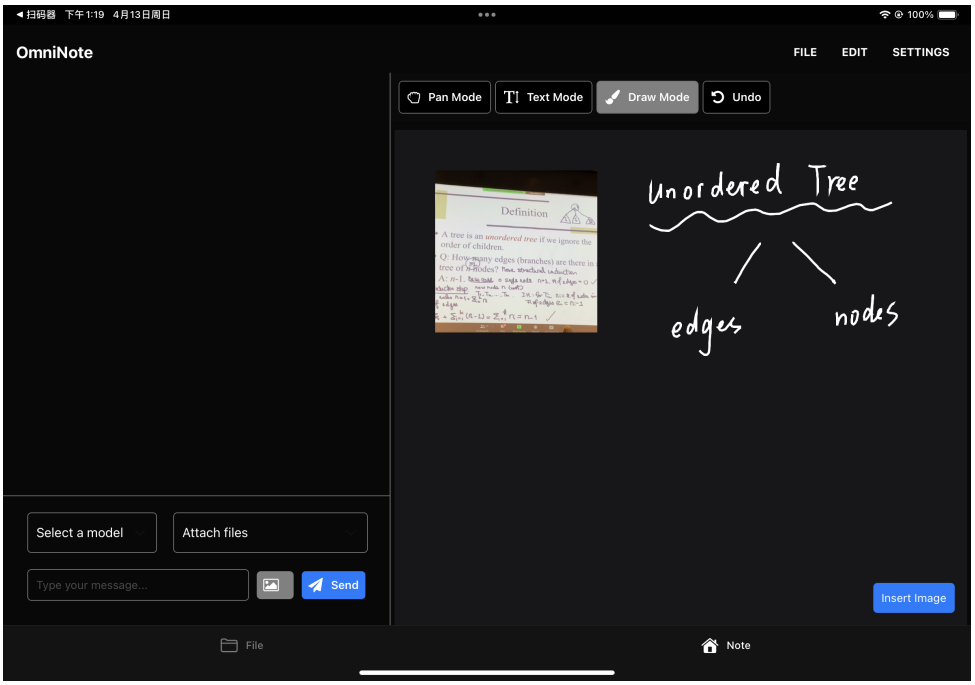


Figure 26: Mobile Application: Note Taking System