ARIAL: An Agentic Framework for Document VQA with Precise Answer Localization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Document Visual Question Answering (VQA) requires models to understand text in complex layouts and ground answers to specific regions of a document image. However, most existing systems prioritize textual accuracy while neglecting spatial grounding, limiting interpretability in high-stakes applications. We present **ARIAL** (Agentic Reasoning for Interpretable Answer Localization), a modular framework that uses a task-oriented planning agent to orchestrate specialized components for OCR, layout parsing, retrieval-augmented generation, and spatial grounding. By decomposing the QA task into structured tool calls, ARIAL enables accurate answer extraction with precise bounding-box localization. We evaluate ARIAL on DocVQA, FUNSD, CORD, and SROIE using both text similarity (ANLS) and spatial metrics (mAP@IoU). ARIAL achieves new SoTA results on all benchmarks, including 88.7 ANLS and 50.1 mAP on DocVQA, surpassing prior systems such as DLaVA. Our results highlight the benefits of modular, interpretable reasoning for document understanding and establish ARIAL as a strong foundation for trustworthy document AI. The coding implementation can be found in: https://anonymous.4open.science/r/ARIAL-26BF

1 Introduction

Document Visual Question Answering (VQA) requires reasoning over both textual content and visual layout in scanned or digitally rendered documents. Models must not only **read and understand** diverse formats—forms, receipts, reports—but also **locate where an answer appears**. This spatial grounding is critical for interpretability, especially in high-stakes domains like finance or law.

While recent models such as LayoutLMv3 Huang et al. [2022], LayoutLLM Luo et al. [2024], and DocLayLLM Liao et al. [2025] have improved textual accuracy by combining language with layout features, they often treat localization as a secondary task. As a result, they may generate plausible answers without clearly identifying their source in the document, making them difficult to verify. Metrics like ANLS Yujian and Bo [2007] capture string similarity but fail to reflect spatial correctness, prompting a shift towards combined evaluations that include IoU for grounding precision.

DLaVA Mohammadshirazi et al. [2024] introduced answer localization to the mainstream by integrating bounding-box prediction within a large multimodal transformer. However, its monolithic design can be **compute-intensive** and may struggle with fine-grained details in dense or handwritten layouts.

In this paper, we propose **ARIAL** (Agentic Reasoning for Interpretable Answer Localization), a modular document VQA framework built around an agentic planning model. Rather than using a single large model, ARIAL delegates subtasks—OCR, layout analysis, retrieval, reasoning, and grounding—to specialized modules orchestrated by a central agent. This agent (implemented with a LLaMA 4 Scout VLM Meta [2025]) dynamically selects tools and composes multi-step reasoning

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

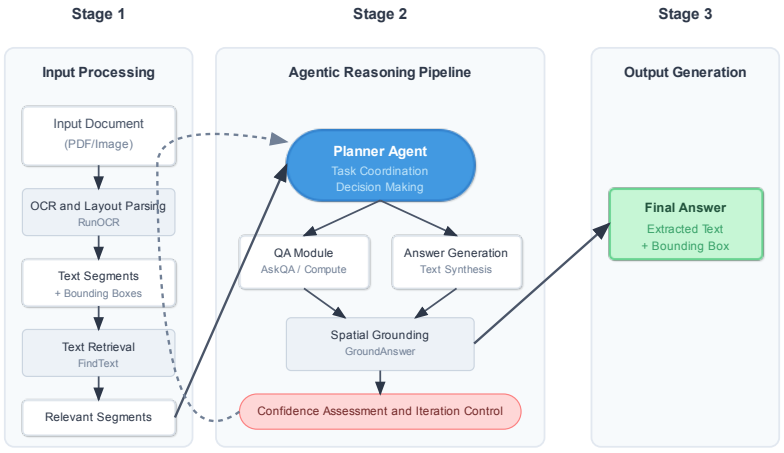


Figure 1: Overview of the ARIAL agentic workflow for Document VQA. The system consists of three modular stages: (1) Input Processing, where an OCR module extracts text segments and bounding boxes from a document image; (2) Agentic Reasoning Pipeline, where the planner agent coordinates task execution—retrieving relevant text, invoking QA or computation, and triggering spatial grounding; and (3) Output Generation, where the final answer and its bounding box are produced. The reasoning loop enables iterative refinement based on confidence, supporting flexible and context-aware decision-making.

chains, enabling accurate and interpretable answers with tight spatial grounding. The key contributions of our work are summarized as follows:

1. **Agentic Document QA**: We introduce the first agent-based document VQA system that decomposes queries into tool calls for OCR, retrieval, and grounding. The modular design enables tool reuse, error tracing, and flexible adaptation across document types.
2. **Precise Answer Localization**: ARIAL produces both the answer text and its bounding box in the image. By aligning answers to OCR-detected spans and contextual cues, it ensures visual traceability and improves trust.
3. **Retrieval-Augmented Reasoning**: To handle long or noisy documents, ARIAL incorporates retrieval-augmented generation Lewis et al. [2020] to focus on relevant text segments, enhancing both reasoning accuracy and efficiency.
4. **State-of-the-Art Results**: On four benchmarks—DocVQA Mathew et al. [2021], FUNSD Jaume et al. [2019], CORD Park et al. [2019], and SROIE Huang et al. [2019b]—ARIAL achieves new best results in both ANLS and mAP@IoU. On DocVQA, it reaches 88.7 ANLS and 50.1 mAP, surpassing DLaVA and other strong baselines.

The rest of the paper is organized as follows: Section 2 reviews related work in document VQA, multimodal models, and agentic AI. Section 3 details ARIAL's architecture and modules. Section 4 outlines datasets and evaluation protocols. Results and analysis appear in Section 5, followed by discussion in Section 6 and conclusions in Section 7.

2 Related Work

2.1 Document VQA and Layout-Aware Models

Early document QA systems treated the task as text-only reading comprehension by applying OCR and feeding the results into standard NLP models Mishra et al. [2019]. However, such approaches

ignored document structure, prompting the development of layout-aware models. LayoutLM Xu et al. [2020], LayoutLMv3 Huang et al. [2022], DocFormer Appalaraju et al. [2021], and StrucTexT Li et al. [2021] embed both text and spatial coordinates to model document layouts more effectively. These models have achieved strong performance on datasets like DocVQA by combining visual and textual representations within unified transformer architectures.

Nevertheless, most of these models output only the answer text and treat localization as an auxiliary prediction or post-hoc mapping. Methods like TILT Powalski et al. [2021] and Donut Kim et al. [2022] further explore end-to-end generation—Donut even bypasses explicit OCR—but still lack transparent mechanisms for spatial grounding. As highlighted by DLaVA Mohammadshirazi et al. [2024], the inability to visualize answer provenance limits model interpretability and hinders error analysis, especially in domains requiring high trust.

2.2 Multimodal LLMs for Documents

Multimodal large language models (MLLMs) such as GPT-4o Hurst et al. [2024], o4 mini high (openai reasoning model) OpenAI [2025], Gemini 2.5 Pro Google Cloud [2025], Claude Sonnet 4 Anthropic, and LLaVA 1.5 Liu et al. [2024] extend VQA capabilities by jointly modeling vision and language. These systems can answer questions directly from document images using prompt-based interfaces. However, they often function as black boxes, lacking explicit reasoning steps and failing to highlight the visual basis of their answers. Their reliance on global visual understanding can lead to errors in fine-grained text recognition and spatial disambiguation Bai et al. [2024].

Recent domain-specific adaptations like DocGPT Iftikhar et al. [2023] and LayoutLLM Luo et al. [2024] augment prompts with structured spatial cues to guide the model's focus. DLaVA Mohammadshirazi et al. [2024] takes a more direct approach by combining detected text with either bounding box metadata (OCR-dependent) or constructed text images (OCR-free), enabling the model to predict both the answer and its spatial location. While DLaVA significantly improves interpretability, it relies on a large, end-to-end multimodal backbone. In contrast, our method adopts a modular agentic design that allows more transparent and controllable reasoning, while retaining compatibility with any OCR or LLM module.

2.3 Agent-Based and Modular Reasoning

Agentic frameworks have recently emerged as powerful alternatives to monolithic models for complex tasks Hayawi and Shahriar [2024]. Systems like HuggingGPT Shen et al. [2023] use a central language model to coordinate multiple tools (e.g., OCR, object detectors, captioning models) for multi-step reasoning. Multi-agent paradigms have been explored for general VQA Zhang et al. [2025], where specialized agents are assigned to subtasks like reading, counting, or visual interpretation. HAMMR Castrejon et al. [2024] introduces a hierarchical architecture that improves reasoning granularity and debuggability.

In the document domain, MDocAgent Han et al. [2025] employs multiple agents for long-document QA, with roles spanning retrieval, modality-specific analysis, key information extraction, and summarization. This modular approach yielded notable gains in performance, demonstrating the potential of agentic decomposition.

Our work builds upon these foundations by tailoring an agentic framework for document VQA. ARIAL leverages OCR tools and layout parsers alongside a planning agent that dynamically routes queries, retrieves relevant content, and grounds answers with bounding-box precision. Unlike generic VQA agents, ARIAL is designed to handle document-specific challenges such as dense typography, noisy scans, and form-based structures. Its modularity allows components—e.g., OCR, retriever, QA model—to be upgraded independently, facilitating efficient adaptation to new domains and improving interpretability.

In summary, ARIAL advances the state of document understanding by combining the zero-shot reasoning power of MLLMs with the transparency and controllability of agentic pipelines. This enables precise answer localization and robust performance across diverse document types.

Table 1: Performance comparison on Document VQA datasets. ANLS measures textual accuracy; mAP@IoU captures spatial localization quality.

2*Method	DocVQA		FUNSD		CORD		SROIE	
	ANLS	mAP@IoU	ANLS	mAP@IoU	ANLS	mAP@IoU	ANLS	mAP@IoU
DocLayLLM (Llama3-7B)	78.4	-	84.1	-	71.3	-	84.3	-
LayoutLLM (Vicuna-1.5-7B)	74.3	-	80.0	-	63.1	-	72.1	-
DLaVA (OCR-Dep)	74.0	34.9	79.6	32.0	82.1	48.0	91.4	-
DLaVA (OCR-Free)	85.9	46.2	87.6	45.5	84.4	57.9	91.4	-
ARIAL (Ours)	**88.7**	**50.1**	**90.0**	**50.3**	**85.5**	**60.2**	**93.1**	**-**

3 Methodology

3.1 Overview

AARIAL is a modular framework for document VQA that employs a reasoning agent to orchestrate specialized tools for accurate answer generation and precise spatial grounding. Figure 1 illustrates the architecture. The central component is a Planner Agent – instantiated by a large language model (we use LLaMA 4 Scout for this role) – which interprets a user query and dynamically routes it through OCR, retrieval, QA, and grounding modules. The agent follows a sense-think-act paradigm, iteratively inspecting intermediate outputs and deciding the next tool invocation based on the query and context.

The complete workflow begins when a document image is provided as input. The Planner Agent first invokes RunOCR(document) to extract text segments with bounding boxes via the OCR module. Next, it calls FindText(keywords) to perform retrieval over these OCR segments, identifying relevant content through the retrieval module. The agent then uses AskQA(context, question) (or performs a Compute() operation for numerical questions) via the QA module to generate a candidate answer. For spatial grounding, the agent invokes GroundAnswer(answer) to pinpoint where the answer appears in the document, using the spatial grounding module. The agent iterates through this pipeline – possibly refining the query, retrieving additional context, or computing aggregate values – until it produces a high-confidence answer and its associated bounding box. A "stop when confident" criterion is used to terminate the reasoning loop once the agent is satisfied with the answer. This flexible multi-step strategy allows ARIAL to adapt the number of tool calls to the complexity of each query. Unlike monolithic end-to-end models, ARIAL explicitly decomposes the task into tool-based subtasks. For instance, if asked "What is the date of the invoice?", the agent may: (1) invoke OCR to extract all text and layout from the document, (2) search the OCR results for date-related terms (e.g., "Date:", or regex patterns matching dates), and (3) extract and localize the date answer. For a more complex question like "What is the total of all items?", the agent can perform a more elaborate sequence: it might retrieve all item prices from the OCR output, use a Compute(sum, values) operation to calculate their total, and then highlight the derived total value in the document image. This step-by-step tool usage yields interpretable reasoning traces that can be inspected. Formally, given a document image I and a natural language question Q, the system must return a textual answer A and a bounding box B_A within I. The agent constructs a sequence of actions $\{a_1, a_2, \ldots, a_n\}$, where each a_i is either:

- a **Tool call**, e.g., `RunOCR(I)`, `FindText(keywords)`, `AskQA(context, Q)`, or `GroundAnswer(answer)`, or
- an **Internal reasoning** step (an intermediate chain-of-thought in the agent's latent state) that guides the next tool selection.

This sequence is dynamically determined by the agent based on the task. The process halts when the agent has produced a confident answer along with its visual grounding. In the following subsections, we describe each module of ARIAL in detail.

3.2 OCR and Layout Parsing

Accurate text extraction is the foundation of ARIAL's reasoning. We employ a two-stage OCR pipeline inspired by DLaVA's approach Liao et al. [2020], but upgraded with a transformer-based

recognizer for improved accuracy. First, a DB (Differentiable Binarization) text detector with a ResNet-50 backbone identifies candidate text regions in the document image, outputting bounding boxes for each text snippet. Then, for each detected region, we apply TrOCR (Transformer OCR) – a state-of-the-art transformer-based text recognition model – to transcribe the text within that region. TrOCR consists of an image encoder and an autoregressive text decoder, leveraging a pretrained Transformer architecture for high-quality OCR (especially on printed and handwritten text). This two-step process yields a set of OCR results: $\{(T_i, B_i)\}_{i=1}^{N}$, where T_i is the text string recognized in region i, and B_i is the corresponding bounding box coordinates in image *I*.

We perform standard preprocessing on input documents to improve OCR performance: scans are scaled to a uniform high resolution (ensuring small fonts are legible), converted to grayscale if necessary, and we apply light noise removal and de-skewing for scanned documents. The OCR module maintains the reading order of extracted text segments (top-to-bottom, left-to-right by default). We also optionally group text segments into structured units (e.g., key-value pairs in forms) using simple layout heuristics or a key-value detection model. The output of the OCR stage is thus a structured text representation with each text element and its bounding box, enabling downstream modules to efficiently look up information by content or by spatial location.

This OCR-driven design prioritizes text fidelity and explicit localization. It is robust on dense documents or fine print where end-to-end vision-language models (VLMs) often fail to read accurately. Although running a dedicated OCR adds slight overhead, it significantly enhances the quality of text extraction and the precision of answer grounding, which is crucial for our task.

Table 2: Ablation Study (DocVQA and FUNSD)

Model Variant	DocVQA ANLS	DocVQA mAP@IoU	FUNSD ANLS	FUNSD mAP@IoU
Full ARIAL (Agent + RAG + GenQA)	88.7	50.1	90.0	50.3
– No Retrieval (all text to QA)	86.2	48.5	88.1	47.9
– Heuristic Agent (no LLM planning)	83.6	44.2	85.4	42.8
– No Generative QA (lookup only)	87.0	49.0	89.0	49.5

3.3 Agentic QA via Retrieval-Augmented Generation

After obtaining OCR results, the agent focuses on the parts of the text relevant to the query. ARIAL employs retrieval-augmented generation (RAG) Lewis et al. [2020] as a step between OCR and answer generation. In particular, the agent's `FindText(keywords)` action performs both lexical search (exact keyword matching) and semantic search over the set of OCR text segments $\{T_i\}$. We implement semantic retrieval by encoding each text segment $\{T_i\}$ into a vector using a Sentence Transformer (MiniLM-v6). The question Q is similarly encoded, and segments with the highest cosine similarity to Q (as well as those containing literal keyword matches) are retrieved. This yields a subset of candidate segments $\{(T_j, B_j)\}$ that are likely relevant to the question.

The filtered text (typically a handful of snippets or lines) is then passed to the QA module. The agent invokes the AskQA tool with the context and question: `AskQA(Context: ..., Question: Q)`. The QA module, implemented with the Gemma 3-27B model, then generates an answer A based on the provided context. Gemma 3 is a 27-billion-parameter generative language model fine-tuned for extractive QA; it excels at reading the provided text and producing a concise answer. Because the QA module is grounded by the retrieved context, ARIAL's answers are less prone to hallucination compared to a generic LLM that sees the whole document.

Notably, the agent can go beyond simple lookup QA with this approach. For example:

- **Computational questions:** If the query requires calculation or aggregation (e.g., "What is the total sum of all items?"), the agent can identify the relevant numeric fields via retrieval, then invoke a `Compute(sum, values)` tool (e.g., summing up the retrieved values) instead of a direct AskQA. The computed result is then treated as the answer candidate.
- **No answer scenarios:** If no relevant text segments are found by retrieval (indicating the answer might not be present in the document), the agent can decide to output a special response such as "No answer found," avoiding unsupported hallucinations.

Throughout this process, the Planner Agent maintains an internal state and chain-of-thought reasoning (not directly shown to the user) which helps it decide which tool to use next. The agent's prompting strategy and policy are designed via few-shot examples of tool use, so that it learns when to call OCR vs. retrieval vs. QA, etc., based on the question. By leveraging these intermediate reasoning steps and tool APIs, ARIAL ensures that each action is transparent and grounded in the document content.

3.4 Spatial Grounding

To localize the answer in the document image, ARIAL uses a grounding module. After the QA step yields an answer text A, the agent invokes `GroundAnswer(A)`. The grounding module attempts to find A (or its components) among the OCR text segments and return the corresponding location B_A.

In the simplest case, if the answer text A exactly matches a substring of some OCR segment T_k, we directly use B_k (the bounding box for that segment) as the answer's bounding box. If the answer spans multiple adjacent OCR segments (for instance, an answer that is a multi-word phrase broken across two lines), we take the union of all involved boxes and merge them into a single bounding region B_A. This provides exact localization of the answer, limited only by the quality of OCR. In practice, because ARIAL's OCR is high-accuracy, this text alignment yields very precise grounding.

For answers that were computed rather than explicitly found (e.g., a sum total), the grounding module highlights the supporting evidence. In these cases, A might not appear verbatim in the text. ARIAL's strategy is to highlight all relevant parts of the document that contributed to the answer. For example, if A is a sum of several values, we will highlight each value that was summed (and possibly an inferred total line if present) to provide a trace of how the answer was obtained.

If the answer text A is somewhat ambiguous (e.g., the answer is "John Doe" but that name appears in multiple places in the document), the agent uses additional context to disambiguate. Specifically, it looks at the context segments retrieved earlier or keywords in the question (e.g., "applicant name") and selects the occurrence of A that is closest to those contextual cues in the layout. This heuristic ensures that the bounding box corresponds to the correct instance relevant to the query.

The final output of ARIAL is a pair (A, B_A), where A is the answer string and B_A is a set of pixel coordinates (or bounding box) in the original image indicating where the answer is located. This spatial output can be visualized for the end-user or used for downstream auditing.

3.5 Training and Fine-Tuning

A key advantage of ARIAL's design is that most components are modular and can be trained or fine-tuned independently. We leverage this to optimize each part of the system:

OCR and Detection: We use pretrained OCR models for both detection and recognition. The DB text detector and TrOCR Li et al. [2023] recognizer were initially trained on large-scale scene text and document datasets, and already exhibit strong performance on our benchmarks. We found no fine-tuning necessary for our tasks, although one could fine-tune these models on domain-specific documents (e.g., handwritten forms) if needed for further gains.

Retrieval Embeddings: For semantic retrieval we use MiniLM-v6 Rao et al. [2025] embeddings (a distilled transformer for sentence embeddings) without additional fine-tuning. Despite being generic, these embeddings proved effective at matching question phrasing to relevant text segments. In future work, one could adapt the embedding model on a QA corpus to further improve query-to-context relevance, but in our experiments, the off-the-shelf model sufficed.

QA Module: We instantiate the QA module with Gemma 3-27B Team [2025], a 27B-parameter language model. We fine-tuned this model on a combination of document QA data (the training sets of DocVQA, CORD, and FUNSD, totaling around 70k QA pairs) using supervised learning. Fine-tuning was done to ensure the model could read the style of text extracted from documents and produce accurate answers. This fine-tuned Gemma 3 model is significantly faster to run than using a 100B+ LLM for QA, making it useful for the bulk of questions. For very simple factoid questions, we observed that even smaller models could suffice, but we opted to use the powerful Gemma 3 to handle the full range of query complexity.

Agent Policy: The Planner Agent is powered by LLaMA 4 Scout. We crafted a set of 50 example question-answer sequences with ground truth tool usage (essentially demonstration traces of how

the agent should behave for various query types). We then fine-tuned the LLaMA 4 model on these traces via behavioral cloning (supervised fine-tuning to mimic the demonstrated sequences). This procedure taught the agent to select tools appropriately and reduced the frequency of redundant or unnecessary actions. We emphasize that this fine-tuning was light (50 demonstrations only) – the majority of the agent's reasoning ability comes from the pre-trained LLM's capabilities, guided by our prompt design and the available tools.

Table 3: End-to-End vs. Agentic Approach Comparison

Metric	LayoutLLM	DocLayLLM	DLaVA OCR-Free	ARIAL (Agentic)
DocVQA ANLS	74.3	78.4	85.9	**88.7**
DocVQA mAP@IoU	–	–	46.2	**50.1**
Average Latency (s/q)	0.7	0.4	1.2	3.2
Interpretability	No	No	Yes	Yes + reasoning trace

4 Experiments

We evaluate ARIAL on four widely-used benchmarks for document VQA and visual information extraction:

4.1 Datasets and Evaluation Metrics

We evaluate ARIAL on four benchmarks: DocVQA Mathew et al. [2021], FUNSD Jaume et al. [2019], CORD Park et al. [2019], and SROIE Huang et al. [2019a], converting form fields or receipt annotations into natural-language questions and treating missing fields as no-answer. We report text accuracy via ANLS (Average Normalized Levenshtein Similarity Yujian and Bo [2007], 0–100%) and localization precision via mean Average Precision over IoU thresholds 0.50–0.95 (mAP@0.50:0.95).

In practice, an ideal system should achieve both high ANLS (answer text is correct) and high mAP (the answer is correctly localized in the document). We therefore track both to evaluate the effectiveness of ARIAL.

4.2 Baselines and Comparisons

We compare ARIAL with a range of layout-aware and LLM-based models:

- **DocLayLLM** Liao et al. [2025] and **LayoutLLM** Luo et al. [2024]: Recent lightweight LLM-based models for document QA. These use LLaMA/Vicuna backbones with prompts that incorporate layout cues (e.g. special tokens or formatting to indicate coordinates or section breaks), often along with chain-of-thought reasoning prompts. We evaluate a publicly available 7B-parameter variant (Vicuna-based) on our tasks.
- **DLaVA (OCR-Dependent) Mohammadshirazi et al. [2024]**: The pipeline introduced in DLaVA, which uses a vision-language model that takes detected text and their positions as input (often rendering text as images or using spatial embeddings) and predicts an answer plus a bounding box. We use DLaVA's OCR-dependent mode as a baseline, which relies on actual OCR text with spatial metadata.
- **DLaVA (OCR-Free) Mohammadshirazi et al. [2024]**: DLaVA's alternative mode, where the system synthesizes visual patches of text (to avoid explicit OCR text inputs) and feeds them into the multimodal model. This was reported to slightly improve performance on some datasets by letting the model implicitly handle text. In our comparisons we include it as it achieved SoTA results on DocVQA and others prior to ARIAL.

4.3 Implementation Details

- **Agent and Models:** In our experiments, ARIAL uses LLaMA 4 Scout (2025 edition) as the Planner Agent LLM by default. This model was fine-tuned on tool use traces as described in Section 3.5.

- **OCR Module:** We implemented the OCR stage using Microsoft's TrOCR model for text recognition. For detection of text regions, we used the open-source implementation of the DB text detector with a ResNet-50 backbone, pretrained on natural scene text and document scans. The combination of DB + TrOCR was chosen for its strong performance on both printed and cursive text (TrOCR has been shown to handle handwriting well, which is critical for datasets like FUNSD). The OCR system operates at 2 seconds per page on average (H100 GPU), which is acceptable given our overall pipeline.
- **Planner Agent Prompting:** The chain-of-thought prompt given to the Planner Agent includes 5 few-shot examples illustrating the sequence of tool calls for various question types. These examples teach the agent how to reason step-by-step (e.g., first OCR, then retrieve, then ask QA, etc.). We constrained the agent's retrieval to a maximum of top-5 segments on DocVQA (which has lengthy documents) and top-3 on the smaller documents (FUNSD, CORD, SROIE) to limit the context size for the QA module and to improve efficiency.
- **QA Module:** The QA sub-module is powered by Gemma 3-27B as mentioned. We fine-tuned Gemma 3 for 3 epochs on the combined document QA training sets (with an Adam optimizer, learning rate 1e-4) to specialize it in extracting answers from provided text. During inference, Gemma 3 processes the retrieved text context (usually a few hundred characters at most) and question, and generates an answer in a seq2seq manner. This model runs comfortably on a single GPU (it uses 20 GB VRAM) and outputs answers in under 0.5 seconds on average.
- **Infrastructure:** All experiments were conducted on a server with 4× NVIDIA H100 80GB GPUs. ARIAL's LLaMA 4 agent runs in 16-bit precision on a single GPU (80GB is sufficient for the 80B model with optimizations), Gemma 3-27B runs on another GPU, and the OCR models run on CPU or a smaller GPU as needed.

5 Results

5.1 Overall Performance

Table 1 presents the performance of ARIAL and baseline models across all four datasets. ARIAL consistently achieves SoTA results on both text accuracy (ANLS) and spatial grounding (mAP@IoU). For instance, on DocVQA, ARIAL attains 88.7 ANLS and 50.1 mAP, significantly outperforming the strongest previous system (DLaVA) which achieved 85.9 ANLS and 46.2 mAP. This is a +2.8 point gain in ANLS and +3.9 in mAP, highlighting the benefit of ARIAL's modular reasoning and fine-grained retrieval. These gains are especially notable given that both ARIAL and DLaVA ultimately rely on OCR-based text spans as input—the difference is that ARIAL's agentic pipeline performs superior answer finding and box selection compared to DLaVA's integrated transformer approach.

5.2 Comparison with Baselines

ARIAL consistently leads over both encoder-only and LLM-based methods. LayoutLLM (Vicuna-1.5-7B) and DocLayLLM (LLaMA-3-7B) achieve 74.3% and 78.4% ANLS on DocVQA with no localization. DLaVA's OCR-Dependent variant records 74.0% ANLS and 34.9% mAP, while its OCR-Free mode reaches 85.9% ANLS and 46.2% mAP. In every case, ARIAL's agentic retrieval, generative QA, and grounding pipeline outperforms these baselines by significant margins, highlighting the effectiveness of explicit tool orchestration.

5.3 Ablation Study

To quantify the contribution of each component in ARIAL, we conduct an ablation analysis on the two primary datasets, DocVQA and FUNSD. Table 2 shows the performance of several ablated variants of our system:

- **Full ARIAL:** This is our complete system (LLaMA4 agent + retrieval + Gemma3 QA + grounding).
- **No Retrieval:** In this variant, we ablate the retrieval-augmented step. Instead of selecting relevant text segments, we feed the entire OCR text (concatenated, truncated to a reasonable

length) to the QA module for answer generation. This causes a drop of 2.5 ANLS on DocVQA and 1.9 on FUNSD, and also slightly hurts mAP. The hit to ANLS confirms that targeted context retrieval is important – giving the QA model too much irrelevant text confuses it and can lead to errors. We also observed this variant is slower, since the QA model must process longer input sequences.

- **Heuristic Agent:** Here we remove the learning component of the agent. Instead of using the LLaMA4 Scout to dynamically decide actions, we script a fixed heuristic pipeline: always do OCR, then retrieval, then QA, then grounding (with no adaptive iteration or reasoning). This simulates an ablative scenario without an intelligent agent. Performance drops substantially (over 5 points ANLS on DocVQA, 4.6 on FUNSD). Many errors in this setting were due to the inability to handle cases where a second round of reasoning or a different tool was needed. For example, if the first retrieval missed something, the heuristic agent would still forge ahead to QA and often output an incomplete answer. This highlights the value of an intelligent planning agent that can adjust its strategy per query and perform disambiguation using spatial reasoning.
- **No Generative QA:** In this variant, we restrict the system to only extract answers by string matching the question against OCR text (similar to a purely retrieval-based QA). If a question asks for a field, we try to find a text snippet following that field's label, rather than using the Gemma3 generator. This degrades ANLS by about 1.7 on DocVQA and 1.0 on FUNSD. The drop, while smaller than other ablations, indicates that the generative QA module contributes to handling the more complex questions where an answer isn't just copyable from the text (for instance, questions that require understanding phrasing or combining multiple pieces of text). The slight hit in mAP likely comes from cases where without the generator, the system might pick a suboptimal snippet as the answer.

6 Discussion

6.1 Interpretability and Trustworthiness

ARIAL's agentic, modular design demonstrates clear advantages over monolithic end-to-end models, as shown by its consistent ANLS and mAP gains across all benchmarks (Table 1). By explicitly orchestrating OCR (TrOCR), retrieval, QA (Gemma 3-27B), and grounding under a LLaMA 4 Scout planner, ARIAL delivers both higher textual accuracy (+2.8–4.4 pp) and improved spatial precision (+3.9–4.8 pp) compared to DLaVA.

6.2 Interpretability and Auditability

Unlike encoder-only approaches (LayoutLLM, DocLayLLM) that lack any spatial outputs, ARIAL produces per-answer bounding boxes. Moreover, the planner's tool sequence (OCR $\rightarrow$ FindText $\rightarrow$ AskQA $\rightarrow$ GroundAnswer) forms a transparent audit trace for every prediction. The ablation study (Table 2) further isolates each component's contribution, guiding targeted debugging when failures occur.

6.3 Modularity and Extensibility

Each module in ARIAL can be upgraded independently: TrOCR can be replaced by future OCR engines; Gemma 3-27B can be swapped for larger or more efficient QA models trained with DORA; and the LLaMA 4 Scout planner can be supplanted by next-generation LLMs. This flexibility—combined with robust gains on forms (FUNSD), receipts (CORD, SROIE), and general documents (DocVQA)—underscores ARIAL's broad applicability.

6.4 Efficiency Trade-offs

As Table 3 shows, ARIAL's multi-stage reasoning incurs higher latency (3.2 s/query) compared to monolithic models (0.4–1.2 s). This cost, however, buys crucial trustworthiness: explicit grounding and reasoning transparency are essential in high-stakes domains where verifying each answer's provenance is a non-negotiable requirement.

7 Conclusion

We introduced **ARIAL**, an agentic framework for Document Visual Question Answering that emphasizes accurate answer extraction and explicit spatial grounding. By orchestrating OCR, retrieval, and question answering through a modular planning agent, ARIAL delivers SoTA performance on multiple benchmarks (DocVQA, FUNSD, CORD, SROIE), surpassing prior methods in both textual accuracy and localization precision.

Beyond performance, ARIAL brings interpretability and flexibility to the forefront. Its modular pipeline enables transparent reasoning steps, tool-level auditability, and adaptability to diverse question types and document structures—capabilities often lacking in monolithic models. This makes ARIAL particularly suited for deployment in high-stakes settings where answer traceability is essential.

Our work demonstrates the potential of agent-driven AI for document understanding by merging the reasoning capabilities of large language models with the precision of specialized vision and OCR tools. This synergy produces a system that not only delivers SoTA performance but also meets real-world demands for trustworthy, explainable AI—positioning ARIAL as a significant advance toward interpretable vision–language systems.

Future Work: We will enhance ARIAL's decision-making through learning-based strategies (e.g., reinforcement learning with rewards for minimal tool use and correct answers) to optimize its querying approach. We plan to broaden its toolkit—adding modules for table parsing and diagram understanding—to tackle complex visual elements like charts and graphs. We also aim to extend ARIAL to multi-document and conversational scenarios, enabling it to synthesize information across documents or engage in follow-up dialogue. Finally, we will conduct user studies to evaluate how visual grounding (highlighted answer regions) influences trust and satisfaction, advancing ARIAL toward seamless integration into daily, document-intensive workflows.

References

Anthropic. Claude Sonnet 4. URL `https://www.anthropic.com/claude/sonnet`.

Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R Manmatha. Docformer: End-to-end transformer for document understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 993–1003, 2021.

Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. Hallucination of multimodal large language models: A survey. *arXiv preprint arXiv:2404.18930*, 2024.

Lluis Castrejon, Thomas Mensink, Howard Zhou, Vittorio Ferrari, Andre Araujo, and Jasper Uijlings. Hammr: Hierarchical multimodal react agents for generic vqa. *arXiv preprint arXiv:2404.05465*, 2024.

Google Cloud. Gemini 2.5 pro, June 2025. URL `https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-pro`. Last updated 2025-06-27 UTC.

Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. Mdocagent: A multi-modal multi-agent framework for document understanding. *arXiv preprint arXiv:2503.13964*, 2025.

Kadhim Hayawi and Sakib Shahriar. Ai agents from copilots to coworkers: Historical context, challenges, limitations, implications, and practical guidelines. *Preprints*, 10, 2024.

Wen Huang, Minghui Qiao, Cong Bai, Yulin Yong, Sheng Zhang, and Qun Guo. Sroie: Scanned receipt ocr and information extraction. In *Proceedings of the ICDAR 2019 Competition on Scanned Receipts OCR and Information Extraction*, 2019a.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4083–4091, 2022.

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE, 2019b.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Linta Iftikhar, Muhammad Feras Iftikhar, and Muhammad I Hanif. Docgpt: Impact of chatgpt-3 on health services as a virtual doctor. *Ec Paediatrics*, 12(1):45–55, 2023.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6. IEEE, 2019.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. In *European Conference on Computer Vision*, pages 498–517. Springer, 2022.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33. 2020.

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13094–13102, 2023.

Yulin Li, Yuxi Qian, Yuechen Yu, Xiameng Qin, Chengquan Zhang, Yan Liu, Kun Yao, Junyu Han, Jingtuo Liu, and Errui Ding. Structext: Structured text understanding with multi-modal transformers. In *Proceedings of the 29th ACM international conference on multimedia*, pages 1912–1920, 2021.

Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11474–11481, 2020.

Wenhui Liao, Jiapeng Wang, Hongliang Li, Chengyu Wang, Jun Huang, and Lianwen Jin. Doclayllm: An efficient multi-modal extension of large language models for text-rich document understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4038–4049, 2025.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024.

Chuwei Luo, Yufan Shen, Zhaoqing Zhu, Qi Zheng, Zhi Yu, and Cong Yao. Layoutllm: Layout instruction tuning with large language models for document understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15630–15640, 2024.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209, 2021.

Meta. Llama 4 Scout, 2025. URL `https://www.llama.com/docs/get-started/`. Large language model, version released April 5, 2025.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 947–952. IEEE, 2019.

Ahmad Mohammadshirazi, Pinaki Prasad Guha Neogi, Ser-Nam Lim, and Rajiv Ramnath. Dlava: Document language and vision assistant for answer localization with enhanced interpretability and trustworthiness. *arXiv preprint arXiv:2412.00151*, 2024.

OpenAI. o4-mini: The latest small o-series model. `https://platform.openai.com/docs/models/o4-mini`, 2025. Accessed: [Current Date].

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. Cord: a consolidated receipt dataset for post-ocr parsing. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019.

Rafał Powalski, Łukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michał Pietruszka, and Gabriela Pałka. Going full-tilt boogie on document understanding with text-image-layout transformer. In *Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II 16*, pages 732–747. Springer, 2021.

Arjun Rao, Hanieh Alipour, and Nick Pendar. Rethinking hybrid retrieval: When small embeddings and llm re-ranking beat bigger models. *arXiv preprint arXiv:2506.00049*, 2025.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

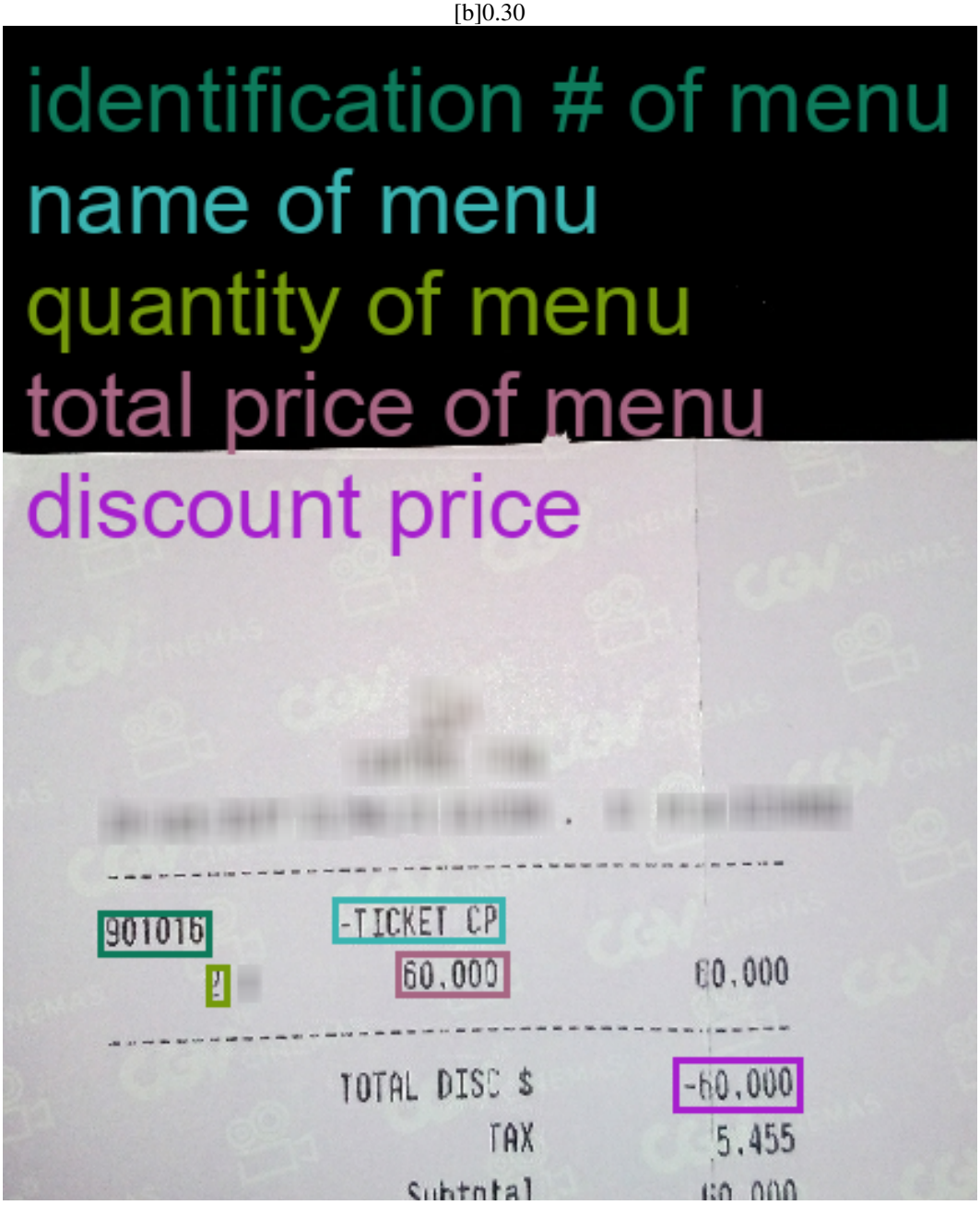
Gemma Team. Gemma 3 technical report. *OpenReview*, 2025. URL `https://openreview.net/forum?id=s8DN0SFkTX`.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1192–1200, 2020. doi: 10.1145/3394486.3403172.

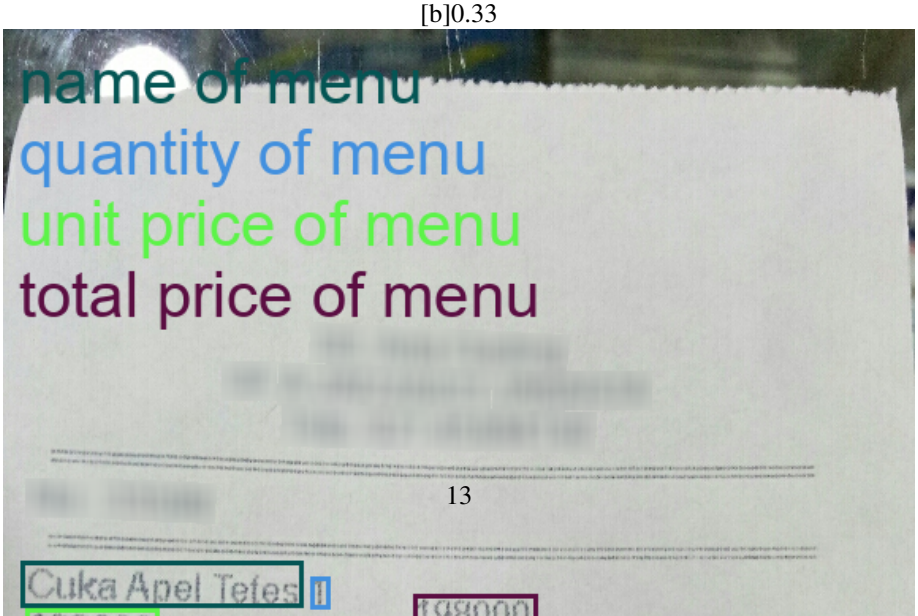
Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095, 2007.

Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, et al. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. *ACM Computing Surveys*, 57(8):1–39, 2025.

[b]0.30


identification # of menu
name of menu
quantity of menu
total price of menu
discount price
901016
-TICKET CP
60.000
60.000
TOTAL DISC $
-60.000
TAX
5.455

[b]0.33


name of menu
quantity of menu
unit price of menu
total price of menu
Cuka Apel Tetes