1 Introduction

OCR is a core component of document processing, which converts scanned or photographed documents into machine-readable text. It supports a wide range of industrial applications, including large-scale digital archiving, information retrieval, and automation of business processes [1] citazioni. Beyond standalone transcription, OCR also provides the foundation for advanced document understanding workflows, allowing downstream tasks such as named-entity recognition, key-value extraction, table parsing, and document-level question answering citazioni. Ensuring high accuracy at the OCR stage is critical, as transcription errors can propagate and compromise the reliability of subsequent tasks [2]. qua sistemo le citazioni e tutto quanto dopo aver finito le sezioni perchè magari modifichiamo l'intro

Recent years have witnessed substantial progress: traditional OCR systems, built around modular detection and recognition architectures, have become highly optimized, while multimodal large language models (LLMs) have emerged as an alternative paradigm, embedding implicit OCR capabilities within end-to-end architectures. These contrasting approaches introduce different trade-offs. Modular OCR systems offer robust handling of scanned documents; multimodal LLMs simplify pipeline design and generalization, but at higher computational cost and latency.

This work conducts a systematic comparative evaluation of 14 solutions, categorized into four groups, as illustrated in Figure 4: (i) open-source OCR engines, (ii) commercial cloud-based OCR services, (iii) commercial multimodal LLMs, and (iv) open-source multimodal LLMs. With the growing prominence of LLMs, this work aims to benchmark the text recognition capabilities of specialized and general-purpose multimodal models against specialized OCR solutions. The evaluation focuses on raw text recognition, without pre- or post-processing, across three industrially relevant use cases: printed receipts, noisy scanned forms, and cursive handwriting. As shown in Figure 1, these scenarios encompass diverse visual and structural conditions, testing various aspects of OCR robustness. Performance is assessed using character accuracy and word accuracy metrics.

In addition to accuracy, this study evaluates inference speed and operational costs, critical factors for real-world deployment. OCR benchmarking is an active research area; however, as discussed in Section 3, previous studies address only specific aspects, whereas a unified evaluation across the four system families considering accuracy, latency, and cost is still missing.

The main contributions of this work are:

- A systematic comparative evaluation of text recognition performance, encompassing both open-source and commercial specialized OCR engines and general-purpose multimodal LLMs.
- A multi-dimensional analysis considering accuracy, inference speed, and operational cost, with direct implications for industrial deployment.
- An evaluation based on three diverse, publicly available datasets, representative of real-world document processing scenarios.
- The public release of evaluation code to ensure reproducibility and facilitate future benchmarking and model extensions.

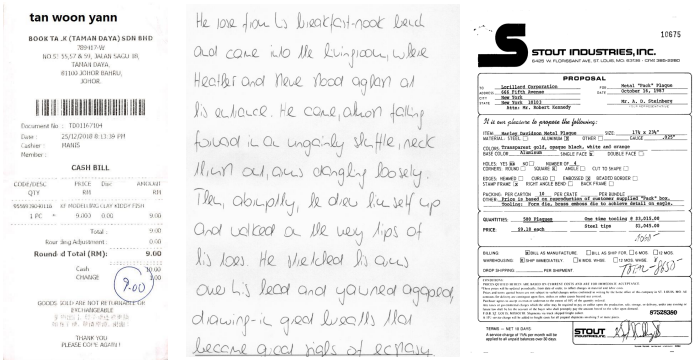


Fig. 1: Examples from the evaluation datasets: printed receipts (SROIE), cursive handwriting (IAM), and noisy scanned forms (FUNSD).

The paper is organized as follows. Section 3 reviews related work. Section 2 provides the technical background, and Section 4 describes the evaluated system categories. Section 5 introduces the methodology and datasets, while Section 6 presents the assessment results, and Section 7 summarizes the conclusions of the assessment.

2 Background

This section introduces the technical foundations of traditional OCR system architectures and multimodal LLMs.

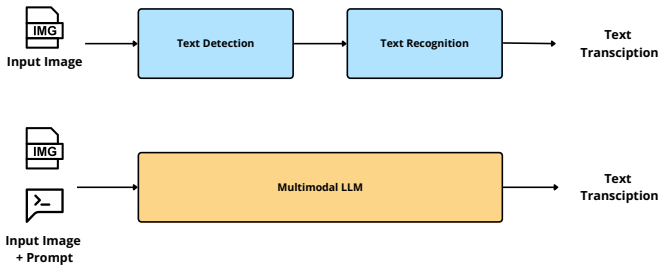


Fig. 2: Comparison between traditional OCR systems and multimodal LLM-based approaches.

2.1 Traditional OCR systems

Traditional OCR systems typically adopt a modular architecture composed of two main stages [3]. The first stage, *text detection*, identifies regions of interest within the document by localizing bounding boxes containing textual content. Early methods relied on connected components or thresholding [4–6] despite being computationally efficient, they proved unreliable when applied to complex inputs such as handwritten text [7]. The advent of deep learning substantially improved the performance of this step. Modern systems leverage convolutional neural networks (CNNs) [8], with detectors such as DBNet [9] achieving strong performance even on curved or irregular text. For latency-sensitive scenarios, lightweight architectures such as MobileNetV3 [10] provide competitive trade-offs between accuracy and efficiency.

The second stage, *Text Recognition*, transcribes the detected regions into character sequences. This task has evolved through several paradigm shifts. Early systems employed handcrafted features and statistical models, including Hidden Markov Models [11]. Again, building on the deep learning paradigm, convolutional and recurrent networks trained with Connectionist Temporal Classification (CTC) loss [12] enabled OCR systems to achieve higher accuracy also in sequence transcription. CRNN [13] is a representative example, combining CNN encoders with recurrent LSTM layers to perform alignment-free transcription. More recently, Transformer-based architectures have become dominant, achieving robust performance across fonts, languages, and complex layouts through self-attention and large-scale pretraining [14]. Among these, lightweight approaches such as SVTR [15] eliminate recurrence in favor of convolutional or mixing-based modules to improve efficiency.

Pre- and Post-processing

Traditional OCR engines relied on explicit pre- and post-processing to handle noise and layout variability [3, 4]. Common steps included binarization and deskewing [5, 16], as well as dictionary or language model–based correction [17, 18], particularly in historical corpora [19].

In constrast, modern deep learning systems embed robustness directly within the model itself, leveraging data augmentation (e.g., rotations, cropping, blur, distortions) and normalization during training to enhance generalization [13, 14]. Recent studies further demonstrate that detectors developed for "in-the-wild" text scenarios can achieve strong results on document datasets once fine-tuned, eliminating the need for handcrafted preprocessing [20].

OCR Fine-tuning

Pre-trained OCR models achieve robust performance on general-purpose documents, but domain adaptation is often required to reach optimal accuracy in specialized contexts. These models typically rely on large-scale and diverse datasets for pretraining, while smaller domain-specific corpora are employed to adapt models to particular fonts, languages, or layouts [21]. Fine-tuning enables this adaptation by specializing components of the pipeline; in modular systems, it can be applied selectively to either detection or recognition. In this work, we focus on the recognition stage, as it is generally less data-intensive and computationally demanding. As illustrated in Figure 3,

OCR datasets provide both bounding boxes for text detection and transcriptions for recognition, which together enable adaptation of both stages.

9

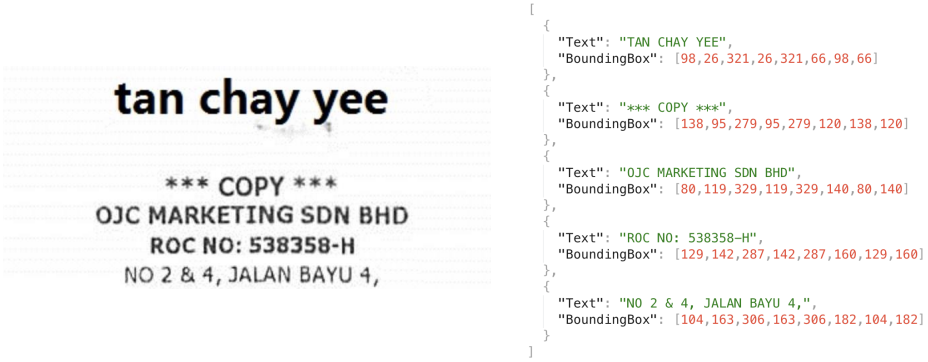


Fig. 3: Example of OCR dataset annotations. Bounding boxes define text regions for detection, while corresponding transcriptions enable recognition. Such annotations support both large-scale pretraining and domain-specific fine-tuning.

2.2 Multimodal LLMs

More recently, multimodal LLMs have emerged as an alternative paradigm for document processing and have rapidly gained traction in industrial applications. In the context of OCR, these models process document images holistically by integrating vision encoders with text decoders in unified architectures [22–24]. This design eliminates the need for separate detection and recognition stages, as illustrated in Figure 2, and enables direct transcription from document images prompted through natural language instructions.

Beyond transcription, the strength of multimodal LLMs lies in their generalization capabilities. Trained on large-scale image–text pairs, they naturally extend to downstream tasks such as key–value extraction, table parsing, and document-level question answering [25], which explains their rapid adoption in industrial workflows.

However, important trade-offs remain. Unlike traditional OCR engines, multimodal LLMs often lack structured outputs such as bounding boxes, layout metadata, and confidence scores that are critical for reliable integration into production pipelines. While some general-purpose models, including Qwen [25] and Gemini [26], are beginning to incorporate features for structured document understanding, these remain limited and non-standardized. In parallel, specialized multimodal LLMs tailored for document processing are emerging. These vertical solutions aim to bridge the gap by providing the structured outputs and reliability essential for industrial-grade applications.

3 Related Work

This section reviews related work in OCR systems evaluation, with a focus on studies that have compared open-source, commercial, and more recently, multimodal LLMs.

3.1 OCR Benchmarks

One of the earliest and most influential contributions in the field is the *Annual Test of OCR Accuracy* [27], which laid the groundwork for benchmarking OCR systems. Despite being outdated, it remains notable as it measured CER, considered latency, and evaluating multiple document types across varied resolutions. However, it relied on a non-public dataset and focused exclusively on legacy systems without including modern cloud-based solutions or open-source deep learning-based approaches. Other early comparative studies, such as [28, 29], explored both open-source and commercial OCR, including web-based and desktop solutions (e.g., ABBYY FineReader with GUI). However, these works relied on private datasets and tools that lacked APIs or libraries, limiting their reproducibility and industrial relevance. Several papers instead addressed specific OCR use cases. For instance, [30] focused on handwritten digits and characters using desktop tools based on Tesseract, resulting not extensive in terms of solutions and scenarios. Similarly, [31] compared OCR tools across 15 categories of noisy or synthetic images, including online services like Google Docs OCR. While notable, for the breadth of categories considered, the dataset was private and lacked focus on scenarios of industrial interest.

As deep learning gained traction, surveys such as [32] began to map the entire document processing pipeline, reviewing models and datasets across tasks. Study [33] evaluated open-source OCR against Microsoft's service in a robotic process automation (RPA) scenario, highlighting performance trade-offs, though only on a small dataset. Finally, [34] compared Tesseract with commercial cloud services (Amazon Textract, Google Document AI) using noisy inputs in English and Arabic. The study contributed a derived dataset and showed that commercial systems were more noise-tolerant, but is not extensive in terms of assessed systems.

Possiamo concludere evidenziando in generale cosa apportiamo rispetto ai questa categoria di lavori precedenti, es. fine-tuning, run-time, latency, costi Capire se invece mettere nell'intro

3.2 Benchmarking on LLMs as OCR Engines

With the advent of vision-enabled large language models (LLMs), a new line of research has emerged exploring their OCR capabilities. Shi et al. [35] was among the first to evaluate GPT-4V on tasks such as OCR, table extraction, and form understanding. The study relied on a limited sample of 50 examples, including handwriting and mathematical formulas.

Subsequent works introduced larger-scale benchmarks. *OCRBench* [36] and *OCRBench V2* [37] evaluate multimodal models on up to 31 datasets and 23 OCR-related tasks, covering scene text, handwritten mathematics, and multilingual settings. These studies highlight the rapid progress of vision-enabled LLMs but also confirm that dedicated OCR systems still outperform them in transcription accuracy. Both benchmarks

mainly focus on LLM-based solutions and line-level recognition, while excluding commercial OCR services and practical tools such as OCR APIs or open-source libraries. Moreover, they do not consider runtime or inference costs—factors that are critical for real-world deployment.

CC-OCR [38] provides a comprehensive benchmark for evaluating the OCR capabilities of multimodal LLMs across 39 sub-tasks, covering multi-scene, multilingual, and structured text recognition challenges. The benchmark emphasizes accuracy-based evaluation but does not account for runtime or inference cost.

WildDoc [39] instead focuses on document understanding in real-world conditions, comprising over 12,000 manually captured document images affected by factors such as illumination, distortion, and viewpoint variation. It assesses model robustness using accuracy, normalized edit distance, and a consistency score, yet similarly omits runtime and efficiency considerations.

OmniDocBench [40] further extends this line of work by benchmarking multimodal LLMs and pipeline-based systems on comprehensive document parsing tasks, including layout detection, text recognition, table extraction, and formula parsing. Although highly extensive, it focuses exclusively on accuracy-based assessment and does not address runtime or computational cost.

Overall, existing benchmarks primarily emphasize accuracy or coverage, overlooking practical factors such as inference time, computational cost, and the inclusion of commercial systems widely used in industry.

4 Evaluated Systems

The two paradigms outlined in Section 2 (modular OCR systems and holistic multimodal language models) are reflected in the systems evaluated in this study. These include both open-source frameworks and commercial services designed for document transcription. In total, 14 systems were evaluate, organized into four categories, as shown in Figure 4: (i) open-source OCR engines, (ii) commercial cloud-based OCR services, (iii) commercial multimodal LLMs, and (iv) open-source multimodal LLMs. These systems were selected for their widespread adoption, accessibility, and suitability for a broad range of document processing tasks, spanning from established open-source solutions to production-grade services.

4.1 Open-Source OCR Engines

Open-source OCR engines are pivotal in academic research and industrial applications due to their transparency, permissive licensing, and adaptable deployment options. These features make them particularly suitable for privacy-sensitive environments. However, achieving optimal performance in production often demands domain-specific expertise and careful optimization. In this study, we evaluated three widely adopted open-source OCR frameworks, each representing a distinct architectural approach as summarized in Table 1. The evaluated frameworks are:

- **Tesseract** [5], a widely used engine maintained by Google. Text detection is performed implicitly through page layout and component analysis, while recognition

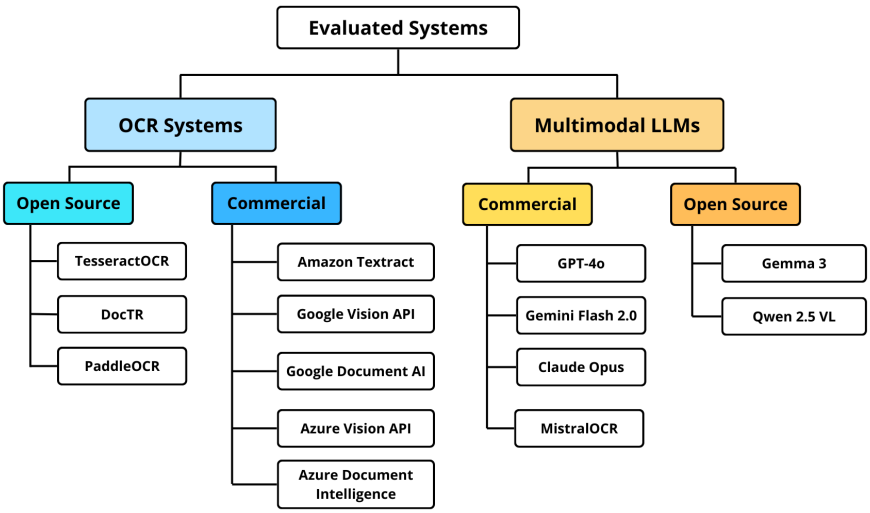


Fig. 4: Overview of the evaluated OCR and LLM-based systems.

relies on an LSTM-based model [41] trained with CTC loss. In our experiments, we employed version 5.0 with default settings through the `pytesseract` interface.

- **DocTR** [42], a modular deep learning library offering pretrained models for both text detection and recognition. In our setup, detection was performed using DBNet with a ResNet-50 [43] backbone, paired with a CRNN recognizer based on VGG16-BN [44]. Experiments were conducted using the official `doctr` Python package.
- **PaddleOCR** [15], an industrial-grade system built on the PaddlePaddle framework. In our experiments, we adopted the lightweight PP-OCRv3 configuration, which combines a MobileNetV3-based detector with an SVTR-LCNet recognizer, and performed inference using the `paddleocr` library.

Table 1: Open-source OCR engines evaluated in this study.

OCR Engine	Version	Detection Stage	Recognition Stage	GPU Support
Tesseract	5.0	Default (PSM=3)	LSTM-CTC (eng)	No
DocTR	0.8.1	DBNet (ResNet-50)	CRNN (VGG16-BN)	Yes
PaddleOCR	2.7	MobileNetV3	SVTR-LCNet	Yes

DocTR and PaddleOCR both support GPU acceleration and were evaluated on CPU and GPU, while Tesseract was tested only on CPU.

4.2 Commercial OCR Service

Commercial OCR services, offered as fully managed cloud platforms by leading providers, deliver scalability, reliability, and minimal setup effort. For this study, we selected services from major cloud providers (Microsoft Azure, Google Cloud, and Amazon Web Services) due to their widespread adoption and support for OCR on both natural images and document-centric inputs.

- *Microsoft Azure* provides OCR through two complementary services, both evaluated in this study. **Azure AI Vision**[1] (version 4.0) is optimized for general-purpose image scenarios and exposes a synchronous API suitable for embedding OCR in user-facing applications. **Azure Document Intelligence**[2] is tailored for scanned and digital documents and leverages an asynchronous API designed for large-scale intelligent document processing.
- *Google Cloud Platform (GCP)*, similarly, offers two distinct OCR solutions. **Cloud Vision OCR**[3] is optimized for text extraction from diverse image inputs, including photographs and natural scenes. **Document AI OCR**[4] targets document-heavy use cases and provides enhanced parsing capabilities for structured and unstructured inputs. Both services were tested to capture their respective strengths.
- *Amazon Web Services (AWS)* provides OCR capabilities through **Textract**[5], a service specialized in document analysis. Textract supports tasks such as text recognition, key–value pair extraction, and table detection, and includes asynchronous processing features designed for large-scale enterprise workloads.

A practical difference worth noting is that Azure and Google allow users to specify the model or version used for transcription, whereas AWS Textract abstracts versioning details, potentially affecting reproducibility and long-term stability in production deployments. The main characteristics of the evaluated Commercial OCR APIs are summarized in Table 2.

Table 2: Commercial OCR services evaluated in this study.

Provider	Service	Target Use Case	Model/Version Control
Microsoft Azure	AI Vision (v4.0)	General-purpose images	Yes (version selectable)
Microsoft Azure	Document Intelligence	Document	Yes (model selectable)
Google Cloud	Vision OCR	General-purpose images	Yes (model selectable)
Google Cloud	Document AI OCR	Document	Yes (model selectable)
AWS	Textract	Document	No (latest)

[1] https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-ocr
[2] https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/overview
[3] https://cloud.google.com/vision/docs/ocr
[4] https://cloud.google.com/document-ai/docs/overview
[5] https://docs.aws.amazon.com/textract/latest/dg/what-is.html

4.3 Open-Source Multimodal LLMs

We evaluated two open-source multimodal LLMs summarized in Table 3, selected for their modern architectures and ability to run efficiently on a single GPU. This enables in-house deployment with full data control and relatively low computational cost. Both models accept text and images as input and can be configured via prompts for OCR-like tasks. They were chosen for their accessibility through Hugging Face repositories and their relevance to document-processing scenarios.

- **Gemma 3** [45], developed by Google DeepMind, was evaluated in its 4B instruction-tuned variant.[6] It integrates a 400M-parameter SigLIP [46] vision encoder, frozen during training, and operates at a fixed resolution of 896×896 pixels. The model supports a context length of up to 128K tokens and emphasizes efficiency through quantization-aware training and architectural optimizations.
- **Qwen2.5-VL** [25], developed by the Alibaba Qwen team, was evaluated in its 3B instruction-tuned release.[7] It employs a ViT-L [47] multimodal encoder trained with a CLIP-style [48] objective, typically processing inputs at resolutions between 448 and 896 pixels. The model supports context lengths of up to 32K tokens and is designed to produce structured OCR-style outputs, including text transcriptions aligned with bounding boxes.

Both Qwen2.5-VL and Gemma 3 achieve strong performance on multimodal benchmarks such as DocVQA and TextVQA, making them suitable for OCR-like transcription tasks.

Table 3: Open-source multimodal LLMs evaluated in this study.

Model	Organization	Parameters	Vision Encoder	OCR Capabilities
Gemma 3	Google	4B	SigLIP ViT	Plain text
Qwen2.5-VL	Alibaba	3B	ViT-L	Plain text + bounding boxes

4.4 Commercial Multimodal LLMs

Commercial multimodal large language models (LLMs) integrate vision and language understanding to perform tasks on sources such as documents and images, and are typically made accessible via conversational interfaces and APIs. Although not specialized OCR systems, multimodal LLMs can be guided by prompting to produce text transcriptions from images. QRepresentative solutions from leading Generative AI providers were selected and are accessed through APIs; these are summarized in Table 4:

[6] https://huggingface.co/google/gemma-3-4b-instruct
[7] https://huggingface.co/Qwen/Qwen2.5-VL-3B-Instruct

- **GPT-4o** (OpenAI): multimodal general-purpose model trained end-to-end across text, vision, and audio, capable of handling inputs in text, audio, image, and video and producing multimodal outputs. [49]
- **Gemini 2.0 Flash** (Google): multimodal general-purpose model from the Gemini family, optimized for low latency and extended context (up to 1M tokens). It supports text, image, and audio inputs and is equipped with structured output and object-detection capabilities. [26]
- **Claude 3.5 Haiku** (Anthropic): fastest and most cost-efficient model in the Claude 3.5 family, with multimodal capabilities including vision inputs and text outputs. It is optimized for low-latency tasks and demonstrates competitive performance on document understanding benchmarks such as DocVQA [50]
- **MistralOCR** (Mistral AI): OCR-oriented multimodal model tailored for document parsing, designed to extract text, tables, and visual elements from documents. It provides structured outputs (e.g., JSON or Markdown) to preserve document layout. [51]

Table 4: Commercial multimodal LLMs evaluated.

Model	Provider	Version	OCR Capabilities
GPT-4o	OpenAI	2024-05-13	Plain text
Gemini 2.0 Flash	Google	2024-12	Plain Text + bounding boxes
Claude 3.5 Haiku	Anthropic	2024-10-22	Plain text
MistralOCR	Mistral AI	1.2	Structured outputs (JSON/Markdown)

5 Assessment Method

This section presents the evaluation methodology employed in the assessment. The methodology includes the definition of the recognition task, the selection of datasets, the inference setup, the evaluation metrics, and the fine-tuning procedure applied to an open-source OCR baseline.

5.1 Task Definition

We evaluated all systems on the task of full-page text recognition, i.e., transcribing all visible textual content in a document image. Since the outputs of the evaluated systems are heterogeneous, they were converted into a plain text representation containing the predicted character sequence. Systems were assessed directly on the original inputs, without any image pre-processing or output post-processing. For the evaluation, the transcriptions were normalized to a common character set, as detailed in Section 5.4. The procedures used to generate and harmonize the outputs for all systems are documented and available in the project repository[8]

[8] URL

5.2 Datasets

We selected three publicly available OCR datasets that are widely used in research and reflect real-world document processing scenarios. These include SROIE for printed receipts, FUNSD for noisy scanned forms, and IAM for handwritten text. Each dataset provides ground-truth transcriptions at the line level, which we aggregated into full-page text files to enable comparison with the model predictions. All experiments were conducted using the official training and test splits released by the respective dataset authors. Both IAM and SROIE required minor preprocessing of either the input images or the ground-truth labels, as detailed below. A summary of the datasets used in our assessment is provided in Table 5

Table 5: Document OCR datasets used in the evaluation.

Name	Type	Use case	Year	Size (Train/Test)
SROIE [52]	Printed receipts	Expense reports	2019	639 / 361
IAM [53]	Handwritten text	Free-text handwriting	2002	1307 / 232
FUNSD [54]	Printed forms	Noisy scanned forms	2019	149 / 50

- *SROIE* The Scanned Receipt OCR and Information Extraction dataset includes 1,000 scanned receipts collected for the ICDAR 2019 Robust Reading Challenge. It features low-quality prints, non-standard layouts, and frequent scanning artefacts. Ground-truth annotations are provided in plain-text format at line level. Inconsistencies in case sensitivity were resolved by normalizing all transcriptions to lowercase.
- *FUNSD* The Form Understanding in Noisy Scanned Documents dataset contains 199 annotated forms extracted from the RVL-CDIP corpus. It is challenging due to low resolution, noise, and diverse layout structures. For the purpose of OCR evaluation, we used only the word-level transcriptions and bounding boxes to generate full-page text files for each document.
- *IAM* The IAM Handwriting Database consists of 1,539 handwritten pages produced by 657 writers. It includes over 115,000 word instances, split into train, validation, and test sets with no writer overlap. Each page contains a printed prompt followed by the handwritten text. Using the official annotations, we cropped out the printed region and retained only the handwritten portion, from which full-page transcription files were generated for evaluation.

5.3 Experimental Setup

Experiments were conducted on a dedicated AWS cloud instance (g5.xlarge) equipped with an NVIDIA A10 GPU (24 GB VRAM), 4 vCPUs, and 16 GB RAM, running Ubuntu 22.04 and CUDA 12.2. All API-based services were accessed through requests issued from the AWS Frankfurt region instance, while open-source OCR engines and LLMs were run locally on the same instance.

Commercial OCR APIs

All experiments were conducted using standard pay-as-you-go cloud accounts at the default service tier. Each service was accessed through its official Python SDK, and exactly one request was submitted per document image, without batching or retries. Table 6 reports the endpoint region employed for each service.

Table 6: Commercial OCR APIs Deployment Regions.

Service	**Region**
AWS Textract OCR API	eu-central-1 (Frankfurt)
Azure (Vision and Document Intelligence)	West Europe (Netherlands)
Google Cloud (Vision and Document AI)	?

Commercial LLMs

All commercial multimodal LLMs were accessed via their official APIs, using the latest model releases available in October–December 2024 (e.g., GPT-4o 2024-05-13, Claude 3.5 Oct-2024, Gemini 2.0 Dec-2024). For consistency, all models were prompted with the unified prompt *"Extract all visible text from the document image. Return plain text"*. Where configurable, the temperature parameter was set to 0; otherwise, provider defaults were used. Each image was submitted in a single request through the text+image completion endpoint.

Open-source LLMs

All open-source LLMs were executed locally using the `vLLM` inference engine (v0.4.2) within Docker containers based on `nvidia/cuda:12.2`. We adopted `vLLM` because it is a widely used open-source solution that provides efficient inference and an OpenAI-compatible API, ensuring comparability across models. Moreover, `vLLM` natively supports a wide range of models available on Hugging Face Hub, which facilitated integration and reproducibility in our setup. The models were exposed through the Chat Completions API, and inference was carried out with default settings except for temperature = 0 to improve reproducibility. One request was issued per image, using the same unified prompt as for the commercial LLMs.

Open-source OCR engines

Open-source OCR engines were accessed via their official Python libraries using default configurations and pre-trained weights. Tesseract required the installation of additional command-line tools, DocTR was executed with the PyTorch backend, and PaddleOCR was run through the PaddlePaddle deep learning framework. Predictions were generated independently for each input file. While all engines allow for task-specific parameterization, only the default settings were employed in order to ensure comparability across systems.

All experiments were orchestrated through reproducible Python scripts, available in the released repository.

Fine-Tuning Procedure

PaddleOCR offers a comprehensive toolkit for both training and evaluation, providing flexible configuration files, data preprocessing utilities, and pretrained models for a variety of OCR tasks. To assess the benefits of domain adaptation, we fine-tuned the SVTR-LCNet recognition module of PaddleOCR on each dataset. Ground-truth annotations were reformatted to match the requirements of PaddleOCR's training pipeline, and three separate models were trained: one for SROIE, one for FUNSD, and one for IAM. Fine-tuning was initialized from the official PP-OCRv3 pretrained weights, using the default training configurations provided by the library. Fine-tuning was performed for 50 epochs on the hardware setup previously described.

5.4 Evaluation Metrics

The evaluation of OCR systems requires considering multiple aspects of performance. In this study, we decided to focus on three key factors: transcription accuracy, inference latency, and costs. Together, these metrics provide a balanced and extensive framework for the scientific validity of different OCR solutions, as well as their practical suitability and impact in industrial applications.

Transcription Accuracy

Regardless of the methodology employed, the quality of a model's output is typically the first aspect to be evaluated. In the OCR domain, transcription accuracy is most commonly measured at the character and word levels through the *Character Error Rate* (CER) and *Word Error Rate* (WER), both derived from edit-distance formulations such as the Levenshtein distance [55, 56].

$$CER = \frac{N_{ins} + N_{del} + N_{sub}}{N_{gt\_chars}} \qquad WER = \frac{N_{ins} + N_{del} + N_{sub}}{N_{gt\_words}} \tag{1}$$

Here, N_{gt_chars} and N_{gt_words} denote the total number of characters and words, respectively, in the ground truth text. The terms N_{ins}, N_{del}, and N_{sub} indicate the number of insertions, deletions, and substitutions required (at the character or word level, depending on the considered metric) to align the model's output with the ground truth text.

In this study, the outlined metrics were computed at the page-level and averaged across the test split of each dataset. Calculations were performed using the open-source `ocreval` toolkit,[9] a modern port of the ISRI Analytic Tools for OCR evaluation. To directly capture both the accuracy of individual character transcription and the ability to produce correct word units, we report *Character Accuracy* (CA) and *Word Accuracy* (WA), defined respectively as $1 - \text{CER}$ and $1 - \text{WER}$.

[9] https://github.com/eddieantonio/ocreval

$$CA = 1 - CER \qquad\qquad WA = 1 - WER \tag{2}$$

To ensure consistency across evaluations, outputs were normalized before metric computation. Only characters within a predefined whitelist were retained: uppercase and lowercase Latin letters (A–Z, a–z), digits (0–9), whitespace, and common punctuation symbols (. , ; : ! ? () [] {} @ # - + ' " /). All other characters were discarded.

Latency

Another key operational factor in the assessment of OCR pipelines is their temporal efficiency. In this context, *inference latency*, defined as the time required by a model to generate the output from a given input, directly affects the feasibility of deploying OCR systems in high-throughput and real-time scenarios.

Our evaluation accounts for the different nature of latency across system types: for commercial OCR engines and multimodal LLMs, which are accessed through cloud APIs, latency includes both model inference time and network overhead. In contrast, for open-source systems, executed locally, latency corresponds solely to the model's inference time. For each dataset, we report the average time (in seconds) required by each system to process a single page, providing a clear and comparable measure of their temporal efficiency.

$$Latency = \frac{\sum_i seconds/page_i}{N_{pages}} \tag{3}$$

Here, given a dataset, N_{pages} represents the total number of pages in the datasets and $seconds/page_i$ denotes the time (in seconds) required by the system to process a page i.

Costs

In addition to accuracy and latency, we finally considered the cost of running OCR systems as a third critical factor. Costs tipically encompass both computational resources and monetary expenses, and become particularly relevant when deploying document processing pipelines at scale. In this study, we focused specifically on monetary costs, which are highly dependent on the specific OCR technology employed:

- **Commercial OCR APIs** typically follow a *page-based* pricing model
- **Multimodal LLMs** adopt a *token-based* pricing model, accounting for both the tokens given as input to the model and those generated as output
- **Open-source Solutions** do not incur direct monetary costs; their expense is estimated solely based on the infrastructural costs afforded to run the experiments

6 Results

This section presents the empirical results of all evaluated systems, structured by scenario: printed receipts (SROIE), scanned forms (FUNSD), and free-text handwritten (IAM). Each scenario is analyzed in terms of transcription accuracy, followed by separate discussions on speed and cost.

6.1 Transcription Accuracy

SROIE: Printed Receipts

On the SROIE dataset, open-source multimodal LLMs demonstrated competitive performance, with Qwen2.5-VL achieving the highest character accuracy at 96.03%, outperforming all commercial OCR services and even fine-tuned open-source OCR engines. Commercial OCR solutions, such as Amazon Textract (94.62% character accuracy, 96.60% word accuracy) and Azure Document Intelligence (95.68% and 95.64%), excel in word accuracy, demonstrating robustness in handling structured text and scanning artifacts.

Fine-tuning significantly improved PaddleOCR's performance, increasing character accuracy from 87.83% to 94.44% and word accuracy from 70.24% to 89.80%, highlighting the effectiveness of domain adaptation.

Commercial LLMs, including GPT-4o (90.83% and 95.69%) and Claude (92.47% and 93.29%), delivered solid results, but were generally outperformed by top OCR systems, while MistralOCR (77.60%) and Gemma 3 (70.79%) showed limitations in raw transcription of scanned receipts.

For scanned receipts, our results show that commercial OCR are the most stable category, but open-source engines achieve competitive accuracy and can be considered a viable option. Moreover, the strong results of Qwen highlight that LLMs are already suitable for real-world production scenarios in this task. Results for all evaluated systems are reported in Table 7.

IAM: Free-text Handwriting

The analysis of the results on the IAM dataset highlights the performance of open-source multimodal LLMs, with Qwen2.5-VL achieving the highest character accuracy at 97.11% and word accuracy at 96.33%, surpassing all other categories. Commercial OCR services, such as Azure Vision (93.49% character accuracy, 94.41% word accuracy) and Azure Document Intelligence (93.06% and 94.19%), delivered robust results, demonstrating adaptability to handwritten text complexities.

Commercial LLMs, including MistralOCR (96.54% character accuracy, 95.26% word accuracy), performed strongly, with MistralOCR leading this group.

Also with this dataset, fine-tuning significantly enhanced PaddleOCR, increasing character accuracy from 45.94% to 84.21% and word accuracy from 20.45% to 72.42%. Open-source OCR solutions like Tesseract (51.7% character accuracy, 21.66% word accuracy) and DocTR (57.88% and 21.18%) struggled notably, indicating limitations in handling cursive text without fine-tuning.

Table 7: OCR performance on the SROIE dataset (printed receipts).

Category	Solution	Character Accuracy	Word Accuracy
Open-source OCR	Tesseract	77.18	71.71
	DocTR	93.36	92.04
	PaddleOCR	87.83	70.24
	PaddleOCR-FT	94.44	89.80
Commercial OCR	Azure Vision	93.88	96.14
	Azure Document Intelligence	95.68	95.64
	Google Vision API	88.09	92.93
	Google Document AI	90.66	92.94
	Amazon Textract	94.62	**96.60**
Commercial LLMs	Claude Haiku 3.5	92.47	93.29
	GPT-4o	90.83	95.69
	Gemini 2.0 Flash	90.41	93.79
	MistralOCR	77.60	83.32
Open-source LLMs	Qwen2.5-VL (3B)	**96.03**	95.08
	Gemma 3 (4B)	70.79	71.36

These findings suggest that for handwritten documents, commercial multimodal LLMs and OCR APIs offer the most effective solutions. Table 8 reports results across all systems.

Table 8: OCR performance on the IAM dataset (handwritten text).

Category	Solution	Character Accuracy	Word Accuracy
Open-source OCR	Tesseract	51.70	21.66
	DocTR	57.88	21.18
	PaddleOCR	45.94	20.45
	PaddleOCR-FT	84.21	72.42
Commercial OCR	Azure Vision	93.49	94.41
	Azure Document Intelligence	93.06	94.19
	Google Vision	88.88	88.89
	Google Document AI	89.00	89.15
	Amazon Textract	93.10	87.34
Commercial LLMs	Claude Haiku 3.5	94.78	93.62
	GPT-4o	95.25	91.61
	Gemini 2.0 Flash	94.58	89.99
	MistralOCR	96.54	95.26
Open-source LLMs	Qwen2.5-VL (3B)	**97.11**	**96.33**
	Gemma 3 (4B)	94.46	91.24

FUNSD: Scanned Forms

On noisy scanned documents, Commercial OCR services provided the most reliable performance. Amazon Textract achieved the highest character accuracy (84.56%), while Azure OCR led in word accuracy (88.15%). These results confirm the robustness of cloud-based pipelines for form-like documents.

Among open-source OCR engines, DocTR offered the best balance (82.42% character accuracy, 79.82% word accuracy), outperforming both the PaddleOCR baseline and its fine-tuned variant.

Observe that in this case fine-tuning did not yield improvements, likely due to the limited size of the FUNSD training set (only 150 annotated images), which restricted the effectiveness of domain adaptation.

Commercial multimodal LLMs performed more modestly than on SROIE. GPT-4o reached 82.85% character accuracy, remaining competitive but still below the strongest OCR APIs. Claude and Gemini produced slightly lower scores, while MistralOCR struggled significantly (58.43%), underscoring the difficulty of directly transcribing structured scanned forms without task-specific optimization.

Open-source multimodal LLMs remained less competitive overall. Qwen2.5-VL achieved 78.97% character accuracy and 82.25% word accuracy, surpassing Gemma 3 but trailing behind specialized OCR systems. Results are summarized in Table 9.

Taken together, the FUNSD results highlight the advantage of dedicated OCR pipelines in structured document scenarios, where multimodal LLMs still face notable limitations.

Table 9: OCR performance on the FUNSD dataset (scanned forms).

Category	**Solution**	**Character Accuracy**	**Word Accuracy**
Open-source OCR	Tesseract	64.87	55.93
	DocTR	82.42	79.82
	PaddleOCR	80.49	68.96
	PaddleOCR-FT	77.03	75.19
Commercial OCR	Azure Vision	84.40	**88.15**
	Azure Document Intelligence	82.74	87.31
	Google Vision API	72.74	81.02
	Google Document AI	72.45	80.82
	Amazon Textract	**84.56**	87.83
Commercial LLMs	Claude Haiku 3.5	74.61	77.80
	GPT-4o	82.85	83.44
	Gemini 2.0 Flash	77.13	83.20
	MistralOCR	58.43	62.32
Open-source LLMs	Qwen2.5-VL (3B)	78.97	82.25
	Gemma 3 (4B)	55.98	58.83

6.2 Time Results

To complement accuracy, we evaluated temporal performance by measuring latency for commercial systems and inference time for open-source ones, reporting the mean, median, minimum, and maximum time per page across datasets.

Results are presented separately for commercial and open-source solutions: commercial engines and multimodal LLMs are accessed through cloud APIs, where latency reflects both model inference time and network overhead. On the other hand, open-source systems are executed locally on the same hardware used for all experiments, ensuring controlled conditions, but potentially lacking the optimization of production-grade services.

SROIE

Commercial OCR services are the fastest, with Azure Document Intelligence averaging 0.23 s/page and Google Vision 0.37 s/page. In contrast, multimodal LLMs are significantly slower: Gemini requires about 2.9 s/page, GPT-4 Vision 8.6 s/page, and Claude more than 10.2 s/page.

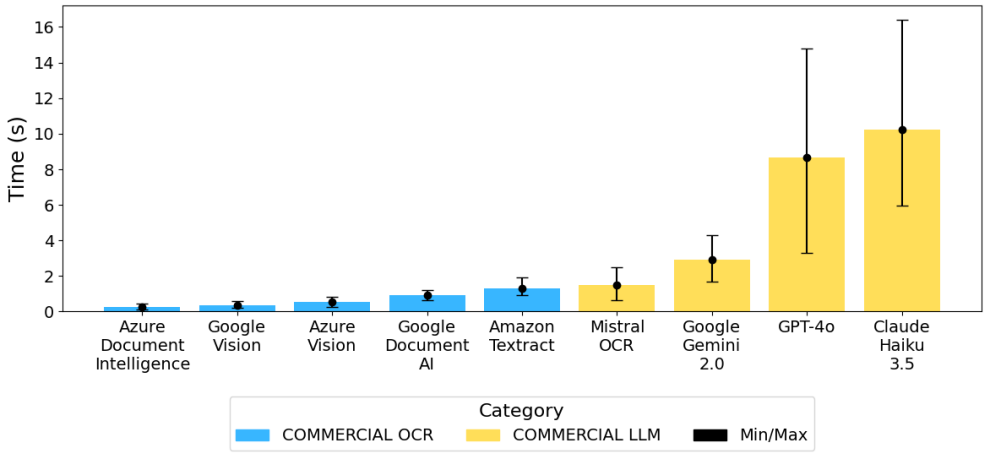


Fig. 5: Mean latency of commercial solutions on SROIE.

IAM

Latencies increase slightly due to higher image resolution, but the ranking remains unchanged. Azure Document Intelligence (0.47 s) and Azure OCR (0.54 s) lead the group, while GPT-4 (7.9 s) and Claude (6.7 s) are an order of magnitude slower.

FUNSD

Results are consistent with the other datasets: Azure Document Intelligence achieves the lowest average latency (0.18 s), Google Vision 0.40 s, while GPT-4 averages 6.7 s and Claude exceeds 10.7 s.

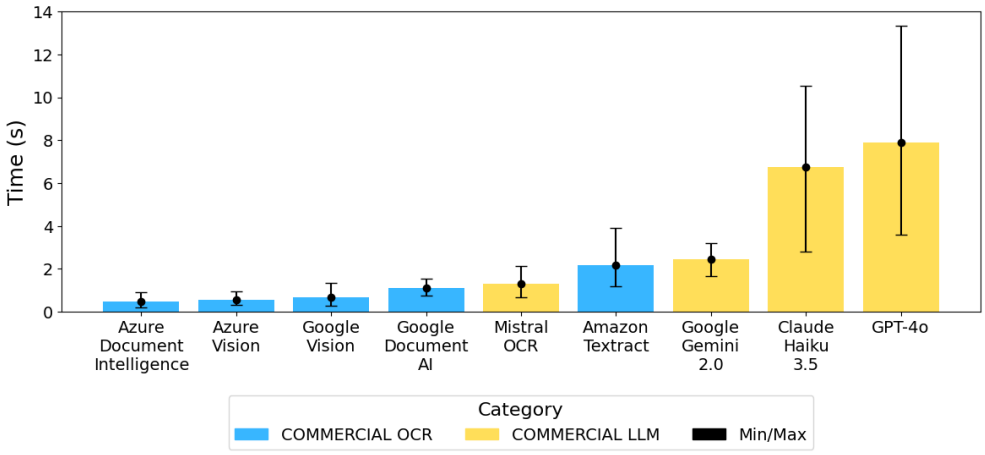


Fig. 6: Mean latency of commercial solutions on IAM.

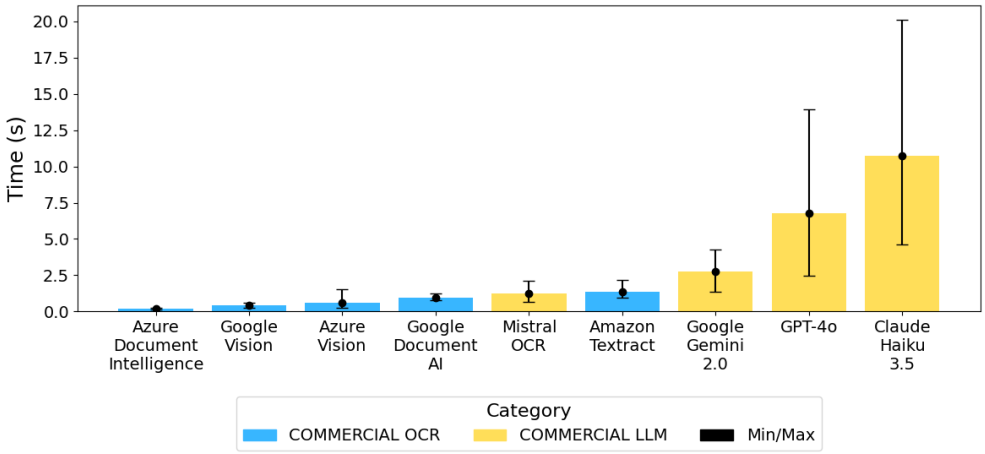


Fig. 7: Mean latency of commercial solutions on FUNSD.

Across all datasets, commercial OCR engines such as Azure, Google, and Textract consistently achieve sub-second latencies, typically processing a page in less than one second. MistralOCR, although a multimodal LLM, shows competitive performance with average latencies around 1.2–1.4 s per page, standing out from the rest of its category. In contrast, frontier multimodal LLMs such as Gemini, GPT-4, and Claude are markedly slower, with per-page latencies between five and forty times higher than OCR engines and often exceeding several seconds. These findings indicate that, despite the accuracy gains offered by LLMs, computational efficiency remains a critical bottleneck, particularly in large-scale or time-sensitive document processing scenarios.

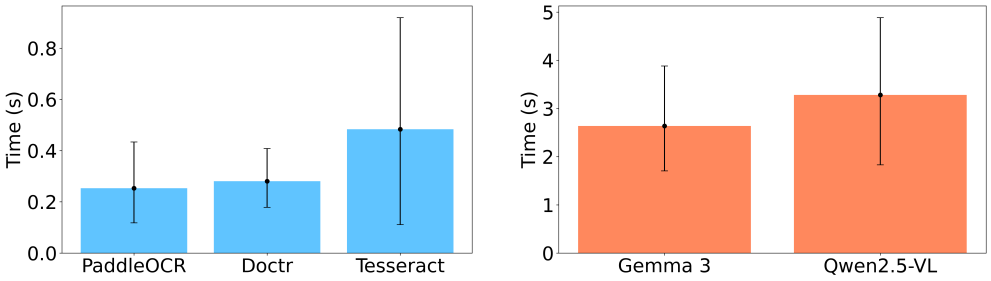
Table 10: Average processing time (seconds per page) across datasets for Commercial systems.

System	SROIE	IAM	FUNSD
Commercial OCR Engines			
Azure Document Intelligence	0.23	0.47	0.18
Azure OCR	0.53	0.54	0.62
Google Vision	0.37	0.66	0.40
Google Document OCR	0.90	1.11	0.95
Amazon Textract	1.30	2.16	1.35
Multimodal LLMs (Commercial)			
MistralOCR	1.49	1.29	1.25
Gemini 2.0 Flash	2.93	2.46	2.74
GPT-4o	8.64	7.91	6.75
Claude 3.5 Haiku	10.22	6.74	10.72

6.2.1 Open-Source Systems

In open-source systems, inference time is influenced by model size, hardware resources, batching strategy, and framework optimizations. OCR engines, being lighter than multimodal LLMs, offer significantly faster predictions. As illustrated in Figure 10a, for SROIE dataset, PaddleOCR achieved the best inference time, followed by DocTR and Tesseract. Even when run on CPU, Tesseract delivered sub-second speed.

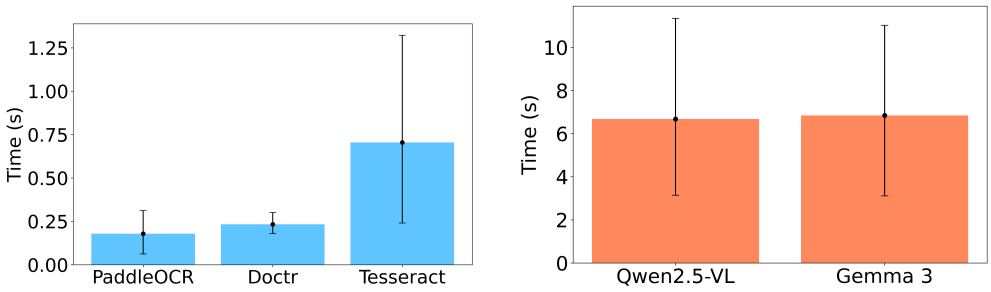
Multimodal LLMs, executed on the same GPU infrastructure, exhibited significantly higher latencies, reflecting their heavier computational load. Figure 8b reports the results. To complement page-level latency, we also measured decoding throughput in tokens per second, a common indicator of autoregressive efficiency. In our experiments, Qwen2.5-VL in its 3B version reached 54.8 tokens/s, while Gemma 3 achieved ???reperisci dato tokens/s. Although these rates are acceptable for offline transcription, they remain far from the responsiveness required in real-time scenarios.



(a) Open-source OCR systems

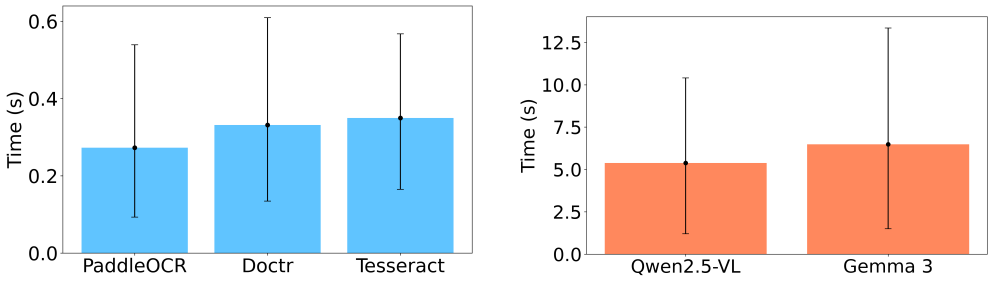
(b) Open-source multimodal LLMs

Fig. 8: Latency comparison of open-source OCR engines and multimodal LLMs on SROIE.



(a) Open-source OCR systems (b) Open-source multimodal LLMs

Fig. 9: Latency comparison of open-source OCR engines and multimodal LLMs on IAM.



(a) Open-source OCR systems (b) Open-source multimodal LLMs

Fig. 10: Latency comparison of open-source OCR engines and multimodal LLMs on FUNSD.

6.3 Costs

Cost is a critical factor in the deployment of document processing systems, particularly when operating at scale. In this section we compare the main cost models associated with the evaluated solutions. Commercial OCR services typically adopt a *page-based pricing* model, while multimodal LLMs are generally billed per token, with different rates for input and output tokens. Open-source systems instead involve infrastructure costs, which depend on the chosen hardware configuration and usage regime.

6.3.1 Page-Based Pricing

Commercial OCR APIs exhibit similar pricing, ranging from $0.001 to $0.0015 per page. MistralOCR is included here as it is the only next-generation solution in our assessment that offers a flat per-page pricing tier.

Overall, Azure OCR and Mistral achieve the lowest costs ($0.63 for all datasets), while the other major providers reporto a cost of $0.94. These differences are negligible at small scale but become relevant when processing millions of pages. It should be

noted, however, that commercial services often apply volume discounts for committed tiers or enterprise contracts that can significantly alter real-world pricing.

6.3.2 Token-Based Pricing

Multimodal LLMs such as Claude, GPT-4o, and Gemini follow a token-based pricing model, with costs scaling according to the number of input and output tokens processed. Each provider applies distinct rates, so we report the aggregate costs incurred per dataset in Table 11.

Results show that Gemini is the most cost-efficient among the evaluated LLMs, with a total of $0.19 across all datasets, while GPT-4o is the most expensive at $2.68. Claude Haiku 3.5 falls in between, with a cost slightly above $1.2. These differences highlight the impact of token pricing on scalability: although per-request costs appear low, they accumulate rapidly with larger volumes, making deployment at industrial scale potentially prohibitive. Moreover, token-based pricing makes cost predictability less straightforward than page-based billing, especially when output length varies significantly between queries.

Table 11: Aggregate costs of multimodal LLMs under token-based pricing.

Provider	FUNSD (50)	IAM (232)	SROIE (346)	Total ($)
Gemini	0.0158	0.0601	0.1106	0.1865
Claude Haiku 3.5	0.1063	0.3779	0.7331	1.2173
GPT-4o	0.2468	0.7326	1.6997	2.6791

6.3.3 Cost of Open-Source Deployment

For open-source systems we estimated deployment costs based solely on infrastructure, assuming all experiments were executed on the same AWS `g5.xlarge` instance reported previously. The on-demand price is approximately $1.006 per hour, corresponding to about $8,800 per year of continuous use. A 1-year Reserved plan reduces this cost to approximately $5,600.

Such costs are sustainable primarily for large-scale processing. Based on our estimates, open-source deployment becomes cost-effective compared to commercial OCR APIs above roughly six million pages per year, and competitive with efficient token-based solutions such as Gemini above twenty million pages per year. These thresholds are high but realistic for industries with massive document throughput, including financial services, insurance, and healthcare.

6.4 Discussion

Manca un po' una conclusione che tiri le somme. Nell'intro è accennata, pensavo fosse sviluppata con maggior dettaglio qui. Forse si può espandare quello che e' in sezione 7?