PaddleOCR 3.0 Technical Report

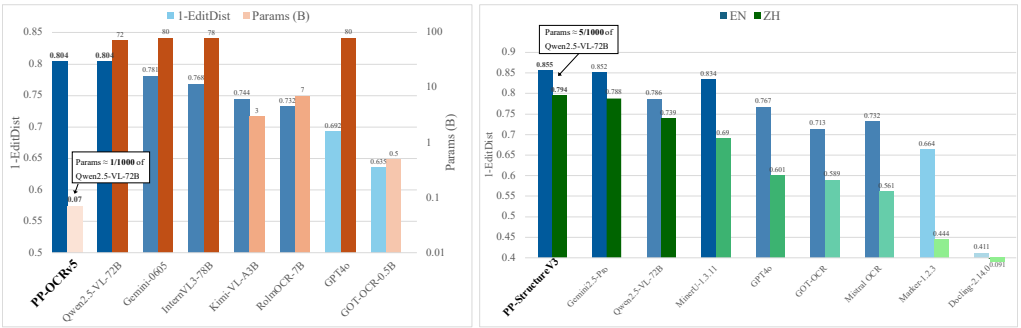
Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, Yanjun Ma

PaddlePaddle Team, Baidu Inc.
paddleocr@baidu.com

Source Code: https://github.com/PaddlePaddle/PaddleOCR
Document: https://paddlepaddle.github.io/PaddleOCR
Models & Online Demo: https://huggingface.co/PaddlePaddle

Abstract

This technical report introduces PaddleOCR 3.0, an Apache-licensed open-source toolkit for OCR and document parsing. To address the growing demand for document understanding in the era of large language models, PaddleOCR 3.0 presents three major solutions: (1) PP-OCRv5 for multilingual text recognition, (2) PP-StructureV3 for hierarchical document parsing, and (3) PP-ChatOCRv4 for key information extraction. Compared to mainstream vision-language models (VLMs), these models with fewer than 100 million parameters achieve competitive accuracy and efficiency, rivaling billion-parameter VLMs. In addition to offering a high-quality OCR model library, PaddleOCR 3.0 provides efficient tools for training, inference, and deployment, supports heterogeneous hardware acceleration, and enables developers to easily build intelligent document applications.



(a) PP-OCRv5 (b) PP-StructureV3

Figure 1 | Performance comparison of PP-OCRv5 and PP-StructureV3 with their respective counterparts. The evaluation set for PP-OCRv5 is our self-built dataset, which includes multiple writing formats such as Simplified Chinese, Traditional Chinese, Chinese Pinyin, English, and Japanese. The evaluation set for PP-StructureV3 is OmniDocBench (Ouyang et al., 2025). The term "1-EditDist" refers to 1 – Edit Distance, where a higher value indicates better performance.

1. Introduction

Optical Character Recognition (OCR) is a foundational technology that enables the conversion of images containing text, scanned documents into structured, machine-readable text. Its significance has never been more pronounced than in the current era of artificial intelligence, where massive volumes of unstructured visual data are generated and consumed daily across scientific, industrial, and social domains. The recent surge in large language models (LLMs)(Achiam et al., 2023; Baidu-ERNIE-Team, 2025; Guo et al., 2025; Yang et al., 2025) and Retrieval-Augmented Generation (RAG) systems(Lewis et al., 2020) has further elevated the strategic importance of OCR: it is no longer sufficient for OCR systems to merely transcribe text accurately—they must now serve as critical enablers in the construction of high-quality datasets, facilitate knowledge extraction, and act as bridges between the visual and semantic layers of modern AI systems.

The evolution of OCR technology reflects the broader trajectory of computer vision and natural language processing. Early OCR systems(Casey and Lecolinet, 1996; Mori et al., 1999), based on hand-crafted features and rule-based heuristics, performed adequately under controlled conditions but quickly reached their limits when confronted with the complexity and diversity of real-world scenarios. The advent of deep learning, particularly convolutional neural networks (CNNs) and their derivatives, ushered in a new era of data-driven OCR, enabling substantial improvements in recognition accuracy, robustness, and adaptability(Goodfellow et al., 2014; Shi et al., 2015). However, as large-scale AI applications proliferate, new requirements have emerged: OCR engines need to handle a broader range of documents—from handwritten notes and multilingual content to rare or historical scripts, and complex layouts with tables, charts, and embedded images. Furthermore, in industrial and research settings alike, OCR is increasingly expected to support downstream tasks such as document understanding, key information extraction (KIE), and semantic search, often as part of end-to-end intelligent workflows.

In recent years, the rapid advancement of LLMs and RAG systems has fundamentally transformed the landscape of information retrieval and knowledge management. These systems rely heavily on the availability of high-quality, diverse, and accurately labeled textual corpora for both pre-training and inference. OCR, in this context, is not simply a data acquisition tool, but a linchpin technology that fuels the entire pipeline—from digitizing vast archives of scientific literature to enabling real-time question answering over heterogeneous document collections. The accuracy and comprehensiveness of OCR outputs directly influence the performance and trustworthiness of LLM-based applications, especially in domains where information is predominantly shared in scanned or image-based formats (e.g., legal documents, historical records, scientific papers, and business forms). Moreover, RAG architectures, which combine retrieval mechanisms with generative modeling, are particularly sensitive to the quality of underlying document representations. Inadequate OCR can propagate errors, introduce noise, or omit critical content, thereby undermining the effectiveness of retrieval and the factual correctness of generated responses.

Despite these pressing needs, existing OCR solutions still face significant challenges in practical deployment. Traditional pipelines typically struggle with low-quality scans, complex backgrounds, non-standard fonts, and multi-modal documents that blend text with figures, tables, or handwritten annotations. The diversity of real-world languages, scripts, and writing styles further complicates the recognition process, requiring not only robust visual modeling but also powerful language understanding capabilities. In addition, industrial and research users increasingly demand lightweight, easily deployable solutions that can be adapted to different hardware constraints and integrated seamlessly with larger AI ecosystems. The open-source community has played a pivotal role in democratizing access to advanced OCR technology,

yet there remains a gap between academic research prototypes and production-ready systems capable of supporting the stringent requirements of dataset construction, RAG workflows, and large-scale document intelligence.

PaddleOCR 1.x & 2.x: Advancements and Innovations in Open-Source OCR Technology

PaddleOCR has emerged as a prominent open-source project addressing these multifaceted challenges. Since its initial release in 2020, PaddleOCR has adhered to the principles of comprehensive coverage, end-to-end workflow, and lightweight efficiency, setting new standards for both usability and technical excellence in the OCR domain. Anchored by the PP-OCR series, PaddleOCR has evolved through multiple iterations—each pushing the boundaries of text detection, recognition, and document analysis. Early versions such as PP-OCRv1(Du et al., 2020) focused on achieving an optimal balance between accuracy and speed, making OCR accessible for resource-constrained environments. Subsequent releases (PP-OCRv2(Du et al., 2021), v3(Li et al., 2022b), and v4) incrementally improved recognition performance, extended language coverage, and introduced sophisticated models for handwriting and rare character recognition. A notable advancement has been the integration of document structural understanding via the PP-Structure series, enabling PaddleOCR to move beyond text lines and paragraphs to address complex layout analysis, table structure recognition (e.g., SLANet(Li et al., 2022a)), and other advanced parsing tasks. These capabilities have made PaddleOCR a critical engine for automated document processing, intelligent archiving, information extraction, and, increasingly, for supporting the data pipelines of LLMs and RAG systems.

The adoption and impact of PaddleOCR in both academic and industrial communities are evidenced by its widespread use and vibrant developer ecosystem. With more than 50,000 stars on GitHub as of June 2025, and its deployment as the core OCR engine in projects such as MinerU(Wang et al., 2024), RAGFlow(KevinHuSh, 2023), and UmiOCR(hiroi sora, 2022), PaddleOCR has become an indispensable tool for digitization initiatives, knowledge management platforms, and AI-driven document analysis workflows. Notably, PaddleOCR has played a central role in the construction of high-quality document datasets for large model training, enabling researchers to assemble diverse, accurately annotated corpora spanning multiple languages, domains, and document types. Its modular architecture and rich API ecosystem facilitate seamless integration with RAG pipelines, where efficient and accurate OCR is essential for document ingestion, retrieval indexing, and context provision to generative models.

As PaddleOCR's user base has expanded, so has the range of feedback and requirements from the community. Users have highlighted persistent needs in areas such as robust handwriting recognition, improved support for multi-language and rare script recognition, more powerful document parsing for complex layouts, and advanced key information extraction. These demands are further amplified by the growing scale and dynamism of LLM and RAG applications, where the ability to extract, structure, and semantically interpret information from diverse documents is a prerequisite for building reliable, responsive, and intelligent systems. Aware of these trends and our responsibility as a leading open-source platform, we remain committed to continuously improving PaddleOCR to meet the evolving challenges of the field.

PaddleOCR 3.0: A New Milestone in Enhancing Text Recognition and Document Parsing

In this context, we introduce PaddleOCR 3.0, a major release designed to systematically enhance text recognition accuracy and document parsing capabilities, with a particular focus on the complex scenarios encountered in modern AI applications. PaddleOCR 3.0 encompasses several core innovations. First, it presents the high-precision text recognition pipeline PP-OCRv5, which leverages advanced model architectures and training strategies to deliver state-of-the-

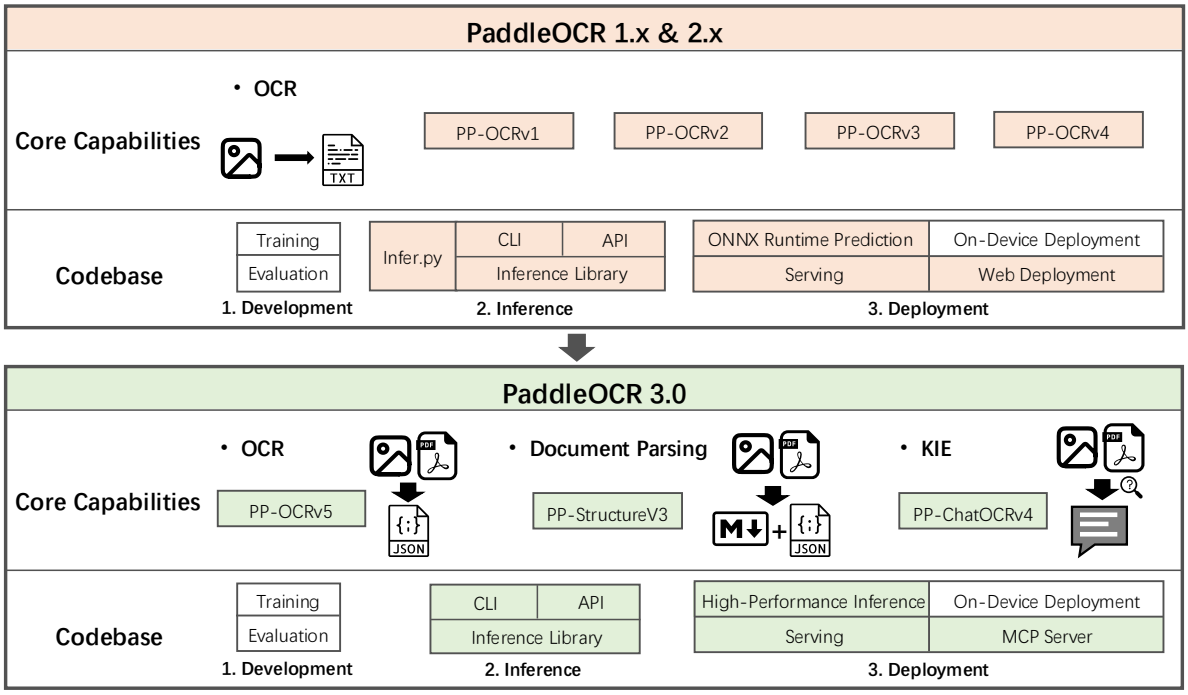


Figure 2 | Evolution from PaddleOCR 1.x & 2.x to PaddleOCR 3.0. Different colors have been employed to denote areas with notable discrepancies between PaddleOCR 1.x & 2.x and PaddleOCR 3.0.

art accuracy across printed, handwritten, and multilingual documents, while maintaining efficiency suitable for both cloud and edge deployment. Moreover, PP-OCRv5 achieves unified recognition of Simplified Chinese, Traditional Chinese, Chinese Pinyin, English, and Japanese within a single model. Second, PaddleOCR 3.0 includes PP-StructureV3, a document parsing solution that integrates layout analysis, table recognition, and structure extraction in an end-to-end framework, enabling accurate and scalable document understanding for forms, invoices, scientific literature, and more. Third, recognizing the need for deeper semantic integration, we introduce PP-ChatOCRv4, a system that combines lightweight OCR models with large language models to facilitate key information extraction, context-aware question answering, and flexible document comprehension—capabilities that are essential for powering RAG pipelines and intelligent document agents. In addition, PaddleOCR 3.0 extends its coverage with dedicated solutions for specialized tasks such as seal text recognition, formula recognition, and chart analysis, further expanding its utility for both research and industrial use cases.

The contributions of PaddleOCR 3.0 are not limited to technical innovation; the release continues to prioritize openness, usability, and extensibility, with a robust API ecosystem, comprehensive model zoo, and active community support. In this version, several legacy design flaws have been revised to provide cleaner and extensible API and CLI, while maintaining a reasonable degree of backward compatibility. At the deployment level, PaddleOCR 3.0 has been restructured to deliver a more streamlined, out-of-the-box experience and improved integration capabilities with LLMs. By targeting key challenges in complex OCR scenarios and aligning with the fundamental needs of data construction for LLMs and RAG pipelines, PaddleOCR 3.0 aspires to serve as an efficient, intelligent, and open infrastructure for document AI. We hope that this advancement will accelerate the development of intelligent automation and knowledge-driven AI systems, foster new research and application frontiers at the intersection of vision

and language, and promote document processing toward higher levels of intelligence and automation.

2. Core Capabilities

PaddleOCR 3.0 comprises three core capabilities: PP-OCRv5, PP-StructureV3 and PP-ChatOCRv4. This section elaborates on the problems addressed by these capabilities, details of the model solution, and their performance metrics.

2.1. PP-OCRv5

PP-OCRv5 is a high-precision and lightweight OCR system designed to perform effectively in a wide range of scenarios. It supports a diverse range of scripts within a single model, including Simplified Chinese, Traditional Chinese, Chinese Pinyin, English, and Japanese. To address the diverse hardware environments and varying requirements for inference speed, PP-OCRv5 offers two distinct model variants: a server version and a mobile version. The server version is specifically optimized for systems equipped with hardware accelerators such as GPUs, thereby enabling accelerated inference and higher throughput. In contrast, the mobile version is tailored for deployment in CPU-only environments, with optimizations targeting resource-constrained devices. Unless otherwise specified, all mentions of PP-OCRv5 in this paper refer by default to the server version. Figure 3 illustrates the framework of PP-OCRv5, which comprises four key components: image preprocessing, text detection, text line orientation classification, and text recognition. The following will introduce these four components.

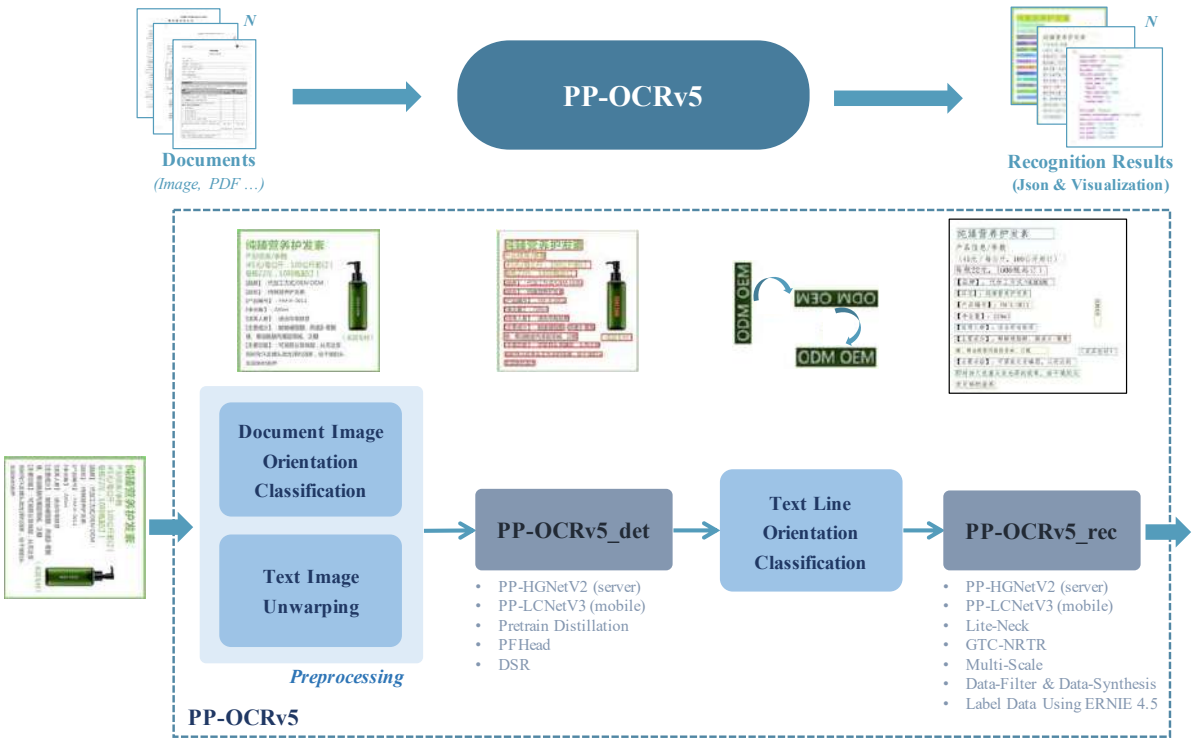


Figure 3 | Pipeline of PP-OCRv5. The pipeline includes image preprocessing, text region detection, text line orientation classification, and text recognition, ultimately extracting the text from images and outputting it as structured textual content.

1. **Image Preprocessing Module**: The image preprocessing module is crucial for preparing the input images by enhancing their quality and adjusting distortions or orientation issues. This process lays a solid foundation for accurate text detection and recognition. PP-OCRv5 includes an optional image preprocessing module to handle image rotation and geometric distortion. This module includes an image orientation classification model based on PP-LCNet (Cui et al., 2021) and a text image unwarping model built on UVDoc (Verhoeven et al., 2023). Users can choose to use these features based on their application scenarios.

2. **Text Detection Model**: The PP-OCRv5 text detection model enhances its predecessor, PP-OCRv4, through optimizations in three key aspects: network architecture, distillation strategy, and data augmentation. Firstly, PP-OCRv5 adopts the more advanced PP-HGNetV2[1] as its backbone network, replacing the previous PP-HGNet. Moreover, PP-OCRv5 enhances model robustness through knowledge distillation, utilizing a visual encoder from the advanced GOT-OCR2.0 (Wei et al., 2024) as the teacher model to transfer knowledge by aligning feature representations with the PP-HGNetV2 student. Finally, PP-OCRv5 detection model enhances text detection performance by incorporating advanced data augmentation techniques, including hard case mining via ERNIE-4.5-VL-424B-A47B comparison and text line-based multilingual strategies (random synthesis, rotation, blurring, geometric transformations). Notably, PP-OCRv5 detection model retains effective strategies from PP-OCRv4, such as the PFHead (Parallel Fusion Head) architecture and the DSR (Dynamic Scale-aware Refinement) training strategy. Overall, PP-OCRv5 improves the model's ability to generalize across diverse datasets, leading to more accurate and reliable text detection in real-world scenarios.

3. **Text Line Orientation Classification Model**: In PP-OCRv5, the text line orientation classification model is primarily responsible for determining and correcting the orientation of detected text lines. When the input text lines are misoriented (e.g., inverted or rotated), this model can automatically identify and rectify their direction to ensure that they are in the standard readable orientation. This step guarantees that the subsequent text recognition model receives a correctly oriented text line, thereby improving the overall accuracy and robustness of the OCR system.

4. **Text Recognition Model**: The PP-OCRv5 recognition model employs a dual-branch architecture with PP-HGNetV2 as the backbone: one branch (GTC-NRTR) uses attention-based training to enhance sequence modeling (Hu et al., 2020), while the other (SVTR-HGNet) focuses on efficient inference with CTC loss. During training, the GTC-NRTR branch guides the SVTR-HGNet branch (Du et al., 2022), but only the lightweight SVTR-HGNet branch is used for prediction, ensuring both accuracy and speed (Li et al., 2022c). For data construction, PP-OCRv5 combines traditional models with the ERNIE-4.5-VL-424B-A47B to automatically annotate and filter high-quality handwritten samples, including rare characters generated through synthesis. Additionally, large-scale labeled data is obtained from documents like PDFs and e-books using automated parsing and edit distance filtering. These data construction strategies also provide a solid data foundation for the overall performance improvement of PP-OCRv5.

Accordingly, the core contributions of PP-OCRv5 are as follows:

1. **Unified Multilingual Modeling**: PP-OCRv5 achieves unified recognition of Simplified Chinese, Traditional Chinese, Chinese Pinyin, English, and Japanese within a single model. Through an innovative unified architecture design, the model maintains a compact size under 100 MB. This resolves efficiency bottlenecks caused by integrating multiple models in multilingual scenarios, significantly simplifying industrial deployment processes. An example

[1] https://github.com/PaddlePaddle/PaddleClas/blob/release/2.6/docs/en/models/PP-HGNetV2_en.md

illustrating PP-OCRv5's multilingual recognition performance is shown in Figure 4.

Figure 4 | Multilingual recognition example.

2. **Robust Recognition of Complex Handwriting**: To address the demands of key application domains such as examination grading in education, bill recognition in finance, and contract entry in the legal sector, PP-OCRv5 has significantly enhanced its capability in handwritten text recognition. Experimental results demonstrate that, compared to previous models, PP-OCRv5 reduces the recognition error rate by 26% on tasks involving non-standard handwriting forms, including both Chinese and English handwritten texts. Figure 5 illustrates the performance of PP-OCRv5 in handwritten text recognition.

Figure 5 | Handwritten Chinese characters (left) and handwritten English text (right).

3. **Robust Recognition of Historical Texts and Uncommon Characters**: In complex and non-standard writing scenarios such as Chinese ancient texts and rare Chinese characters, PP-OCRv5 significantly improves text recognition accuracy through optimization of network architecture and systematic construction of diverse, high-quality datasets, effectively meeting the high-precision text recognition requirements across various domains. The recognition performance in complex scenarios is shown in Figure 6.

We evaluated the OCR capabilities across 17 different scenarios, including handwritten Chinese, handwritten English, printed Chinese, printed English, Chinese Pinyin, Japanese, Chinese ancient texts, traditional Chinese, common, blurred, rotated, Greek characters, emojis, tables, artistic fonts, special symbols, and deformed scenes. These diverse scenarios allow us to comprehensively assess the performance and adaptability of the system. Based on the OmniDocBench OCR text evaluation standards, we conducted extensive testing on mainstream OCR methods and multimodal large models. Figure 7 presents the evaluation results, using 1-edit distance as the metric, and lists the average metrics across all scenarios as well as specific performance metric for key scenarios, including handwritten and printed Chinese and English, Chinese Pinyin, and Chinese ancient texts.

The results indicate that the lightweight PP-OCRv5 ranks first in terms of the average 1-edit

Figure 6 | Vertical Chinese ancient book text (left) and rare Chinese character (right).

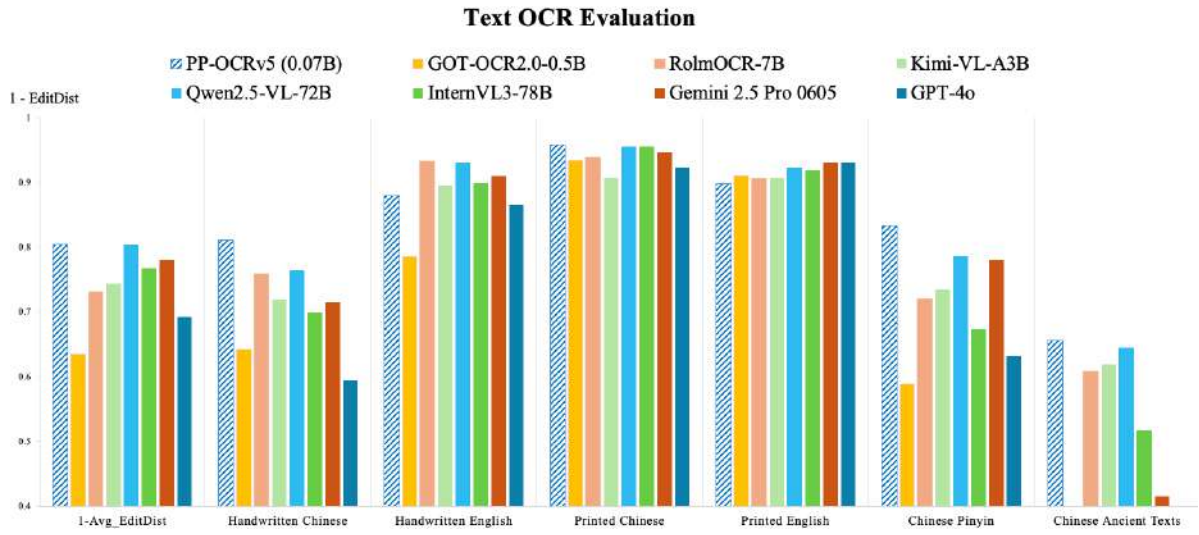


Figure 7 | Text OCR evaluation. The term "1-EditDist" refers to 1 – Edit Distance, a higher value of this metric indicates superior performance.

distance across all scenarios, surpassing all multimodal large models such as GOT-OCR2.0-0.5B (Wei et al., 2024), RolmOCR-7B (AI, 2025), Qwen2.5-VL-72B (Yang et al., 2024), InternVL3-78B (Chen et al., 2024), Gemini 2.5 pro 0605[2], and GPT-4o[3]. In Chinese scenarios, whether handwritten or printed, PP-OCRv5 significantly outperforms other methods. Although slightly inferior to multimodal solutions in handwritten English recognition, it is noteworthy that our model has only 0.07 B parameters. In the Chinese Pinyin and Chinese ancient text scenarios, PP-OCRv5 has achieved significant advantages. These results demonstrate that lightweight models specifically designed for OCR tasks can match or even exceed the accuracy of large-scale multimodal models. Simultaneously, they offer substantially reduced computational and storage requirements, significantly enhancing inference efficiency and deployment flexibility, and thereby delivering critical advantages for industrial applications and mobile device deployment.

[2] https://deepmind.google/models/gemini/pro/
[3] https://openai.com/index/hello-gpt-4o/

2.2. PP-StructureV3

PP-StructureV3 is a multi-model pipeline system developed for document image parsing tasks, it can accurately and efficiently convert document images or PDF files into structured JSON files and Markdown files. As illustrated in the algorithm framework in Figure 8, the system primarily consists of five modules: preprocessing, OCR, layout analysis, document item recognition, and postprocessing.

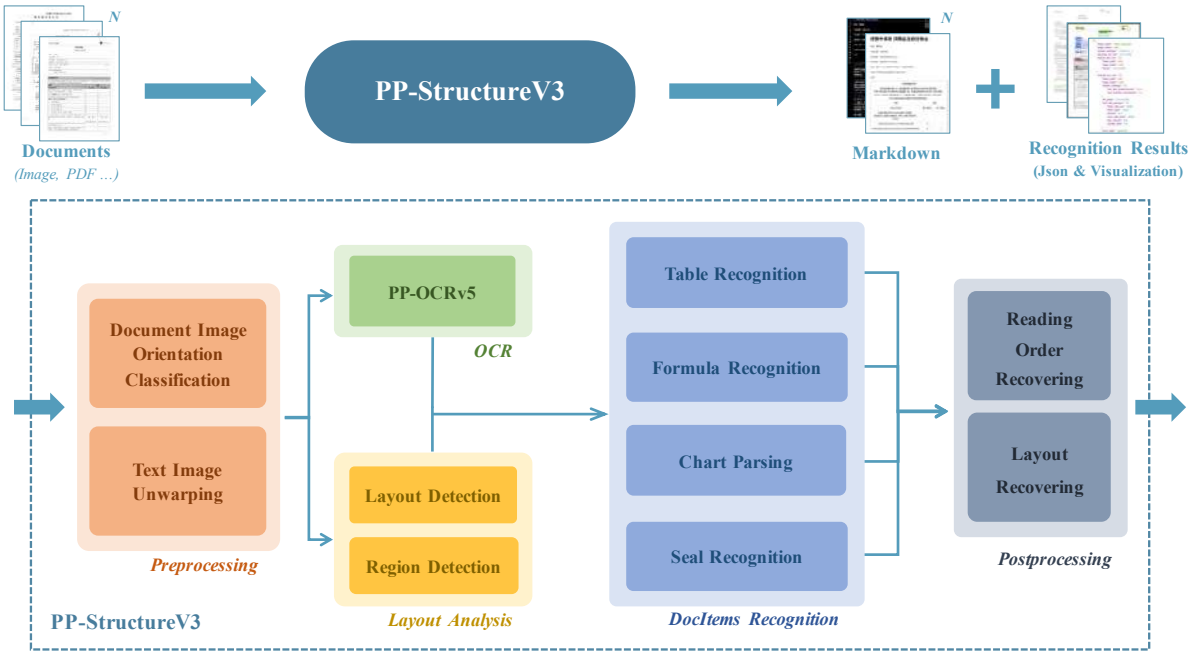


Figure 8 | Pipeline of PP-StructureV3. The pipeline includes Preprocessing, OCRv5, Layout Analysis and Document Items Recognition and Postprocessing. It effectively parses content from images and outputs it as structured data.

1. **Preprocessing**: Similar to PP-OCRv5 (see Section 2.1), this module comprises a document image orientation classification model based on PP-LCNet and a text image unwarping model based on UVDoc. It is primarily designed to address issues related to low-quality document images, such as rotation and distortion.

2. **OCR**: This module employs PP-OCRv5 (see Section 2.1) with preprocessing disabled to detect and recognize all textual content within document images. Compared to PP-OCRv4, PP-OCRv5 achieves significant performance improvements via optimizations in network architecture, training strategies (such as knowledge distillation), and enhancements to the training dataset. Notably, it demonstrates substantial improvements in detection and recognition for challenging scenarios, including vertical text layouts, handwritten text, and rare Chinese characters.

3. **Layout Analysis**: This module incorporates two models: a layout detection model and a region detection model. The layout detection model, PP-DocLayout-plus, is an optimized version of the PP-DocLayout(Sun et al., 2025). It significantly enhances layout detection performance for complex documents, such as multi-column magazines and newspapers, reports with multiple tables, exams, handwritten documents, Japanese and vertically oriented layouts documents. The newly proposed layout region detection model addresses the problem of multiple articles appearing on a single layout page. For example, a single newspaper page often contains several

Method Type	Methods	Edit ↓	
		EN	ZH
Pipeline Tools	PP-StructureV3	**0.145**	**0.206**
	MinerU-1.3.11 (Wang et al., 2024)	0.166	0.310
	MinerU-0.9.3 (Wang et al., 2024)	0.150	0.357
	Mathpix[1]	0.191	0.365
	Pix2Text-1.1.2.3 (breezedeus, 2022)	0.320	0.528
	Marker-1.2.3 (Paruchuri, 2023)	0.336	0.556
	Unstructured-0.17.2 (Unstructured-IO, 2022)	0.586	0.716
	OpenParse-0.7.0 (Filimoa, 2024)	0.646	0.814
	Docling-2.14.0 (Docling Team, 2024)	0.589	0.909
Expert VLMs	GOT-OCR2.0 (Wei et al., 2024)	0.287	0.411
	Mistral OCR[2]	0.268	0.439
	OLMOCR-sglang (Poznanski et al., 2025)	0.326	0.469
	SmolDocling-256M_transformer (Nassar et al., 2025)	0.493	0.816
	Nougat (Blecher et al., 2023)	0.452	0.973
General VLMs	Gemini2.5-Pro[3]	0.148	0.212
	Gemini2.0-flash[4]	0.191	0.264
	Qwen2.5-VL-72B (Yang et al., 2024)	0.214	0.261
	GPT-4o[5]	0.233	0.399
	InternVL2-76B (Chen et al., 2024)	0.440	0.443

[1] https://mathpix.com/
[2] https://mistral.ai/
[3] https://deepmind.google/models/gemini/pro/
[4] https://deepmind.google/models/gemini/flash/
[5] https://openai.com/index/hello-gpt-4o/

Table 1 | Comprehensive evaluation of document parsing methods on OmniDocBench.

distinct articles. Using only the layout region detection model makes it challenging to correctly associate elements with their respective articles, leading to errors in reading order recovery. By introducing the layout region detection model, elements can be accurately assigned to their corresponding articles.

4. **Document Items Recognition**: Based on predictions from the layout detection models, the content of each page element is recognized using appropriate methods, including the table recognition solution PP-TableMagic, the formula recognition model PP-FormulaNet_plus, the chart parsing model PP-Chart2Table, and seal recognition with PP-OCRv4_seal.

- **Table Recognition**: PP-TableMagic is a comprehensive table recognition system composed of several specialized models. It includes a table orientation classification model and a frame type classification model, which determine the rotation and framing style of the table, respectively, guiding the selection of the appropriate recognition method. The cell detection model, based on object detection algorithms, accurately locates individual table cells. Additionally, the structure recognition model outputs the table's structure in HTML format.
- **Formula Recognition**: This model is an enhanced version of PP-FormulaNet (Liu et al., 2025), capable of recognizing images containing formulas cropped from full document

images and generating the corresponding LaTeX code. To address the recognition of complex multi-line formulas, the token length was increased to 2560, and the training dataset was expanded to include more complex formulas. Additionally, to handle formulas containing Chinese characters, a large volume of relevant data was mined for training.

- **Chart Parsing**: PP-Chart2Table is a lightweight, end-to-end vision-language model designed to accurately extract data from various types of chart images, such as histograms, line charts, and pie charts, and convert the extracted information into tables represented in markdown format. It genuinely understands and retrieves chart data through an innovative Shuffled Chart Data Retrieval task and meticulous token masking. Its performance is further boosted by a sophisticated data synthesis pipeline that generates diverse, high-quality training data using RAG with high-quality seeds and LLM persona design. A two-stage LLM distillation process, leveraging large volumes of unlabeled out-of-distribution (OOD) data, enhances the model's adaptability and generalization to real-world scenarios.
- **Seal Recognition**: PP-OCRv4_seal is a system specifically tailored for the recognition of oval, round, and other types of seals. It incorporates a curved text detection model, which can accurately detect and rectify bent text, as well as a general-purpose text recognition model.

5. **Post-processing Module**: Upon completion of the above steps, the positions and corresponding content of each element in the document are obtained. The post-processing module then reconstructs the relationships among elements, such as linking figures and tables with their captions and recovering the correct reading order. PP-StructureV3 improves upon the X-Y Cut (Ha et al., 1995), significantly enhancing the reconstruction of reading order in complex layouts, including magazines, newspapers, and vertically typeset documents.

To evaluate the performance of PP-StructureV3, we conducted experiments using the OmniDocBench benchmark, with the results presented in Table 1. As shown in the table, PP-StructureV3 demonstrates exceptional performance in Chinese and English document parsing, establishing itself as the current SOTA. It not only significantly outperforms other pipeline-based tools, but also shows strong competitiveness when compared to the most popular expert VLMs and general VLMs.

2.3. PP-ChatOCRv4

PP-ChatOCRv4 is an advanced key information extraction solution for document images, leveraging LLMs, VLMs, and OCR technologies to enable robust key information extraction in challenging scenarios such as complex layouts, multi-page PDFs, rare characters, intricate table structures, and documents containing seals. As illustrated in Figure 9, the overall workflow of PP-ChatOCRv4 comprises the following components: the layout analysis module PP-Structure (Li et al., 2022a), the vector retrieval module, the large language model ERNIE 4.5, the document-oriented vision-language model PP-DocBee2 (Ni et al., 2025), and the result fusion module.

1. **PP-Structure**: PP-Structure serves as the document image parsing module. It is built upon multiple specialized models, including layout detection, text line detection, text recognition, and table structure recognition models. By leveraging these models, PP-Structure is able to parse various elements from document images, thereby generating a structured, text-based representation of the document content.

2. **Vector Retrieval Module**: A feature vector database is constructed using the textual content parsed from document images. During key information extraction, RAG technology

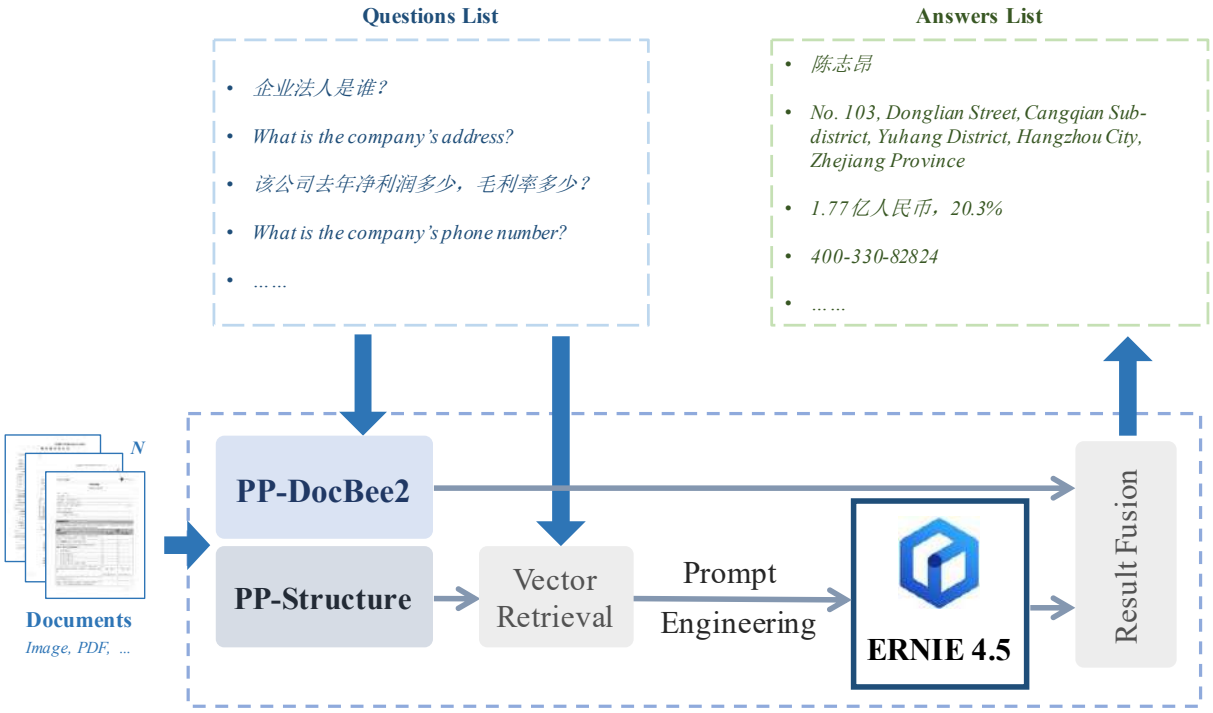


Figure 9 | Pipeline of PP-ChatOCRv4.

is initially employed to efficiently identify and extract critical information from lengthy and redundant texts. By leveraging RAG, the efficiency and accuracy of information retrieval are significantly enhanced.

3. **Large Language Model**: The framework supports any large language model (LLM); for example, we currently use ERNIE-4.5-300B-A47B, the latest LLM released by Baidu, to extract information based on carefully designed prompts. These prompts are crafted to seamlessly integrate retrieved textual information with user queries, thereby enhancing both the efficiency and accuracy of the results.

4. **PP-DocBee2**: The PP-DocBee2 is a novel multimodal large language model with 3 billion parameters designed for end-to-end document image understanding developed by us. It is capable of directly extracting text-based answers from document images using prompts constructed from the given questions.

5. **Result Fusion**: Extraction results from both text-based and image-based approaches are fused to produce the final output.

To evaluate the performance of PP-ChatOCRv4 and other methods, we designed an end-to-end evaluation pipeline using a custom multi-scenario benchmark dataset comprising 638 document images. These images span a wide range of scenarios, including financial reports, research papers, contracts, manuals, regulations, as well as humanities and science papers. Each document image is accompanied by several questions and corresponding answers, resulting in a total of 1,196 question-answer pairs. The results are presented in Table 2.

Methods	Recall@1
GPT-4o	63.47%
PP-ChatOCRv3	70.08%
Qwen2.5-VL-72B	80.26%
PP-ChatOCRv4	**85.55**%

Table 2 | Comprehensive evaluation of various solutions on a custom multi-scenario benchmark: overall recall@1 performance based on ground truth comparison.

3. Codebase Architecture Design

3.1. Overall Architecture

The overall architecture of the PaddleOCR 3.0 codebase is illustrated in Figure 10. At its foundation, PaddleOCR 3.0 is built upon the PaddlePaddle framework (Ma et al., 2019), which incorporates a neural network compiler for performance optimization, supplies a highly extensible intermediate representation (IR), and ensures broad hardware compatibility. Building on this foundation, PaddleOCR 3.0 is structured around two core components: a model training toolkit and an inference library. The inference library offers flexible integration paths that naturally extend to deployment. The deployment capabilities will be introduced in Section 4.

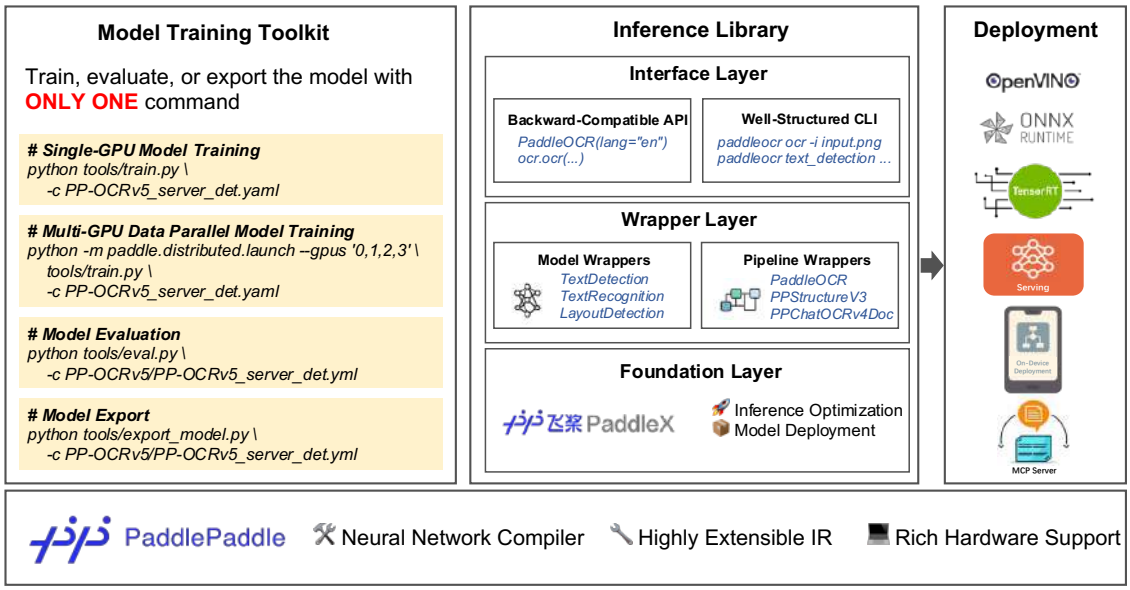


Figure 10 | Overall Architecture of the PaddleOCR 3.0 Codebase.

The training toolkit provides a comprehensive suite of utilities that support the complete training pipeline for various models, including text detection models, text recognition models, etc. It also facilitates the conversion of trained models from dynamic graph format to static graph format, thereby enhancing their suitability for inference and deployment in production environments. Users can execute Python scripts with a single command to perform tasks such as model training, evaluation, and export. Additionally, various parameters can be configured to meet different requirements, such as specifying the path to a pre-trained model or using a custom dataset directory.

Complementing this, the inference library is designed to be lightweight and highly efficient.

It supports loading both officially released inference models and custom models trained by users. The library enables inference across eight end-to-end model pipelines and can be readily integrated into real-world applications. We will elaborate the design of the inference library in the next subsection. The inference library serves as the foundation for downstream deployment capabilities, including high-performance inference across various frameworks, deploying the model pipeline as a service, deploying it to mobile devices, and running it via a Model Context Protocol (MCP) server.

3.2. Inference Library Design

In this subsection, we present the rationale behind the design and structural organization of the PaddleOCR 3.0 inference library.

Let us start with the defects of the inference library in PaddleOCR 2.x:

- All parameters of the PaddleOCR 2.x CLI exist in the same namespace. This is not extensible, as each new feature required manually adding more arguments to an increasingly bloated global parameter list, making it difficult to maintain and scale.
- In PaddleOCR 2.x, the parameters are only configurable through function arguments. Such an approach hinders reproducibility and portability, especially in scenarios where configurations need to be shared or version-controlled.
- PaddleOCR 2.x lacks clear separation of concerns—the boundary between the model development toolkit (primarily designed for training) and the inference library was not clearly defined. Instead, the inference library was built directly on top of the model development toolkit inference scripts, which introduced two entry points for the inference functionality, potentially causing confusion for users. Additionally, the inference library was constructed based on assumptions about what the entry points should be, which broke modularity and maintainability.

To address these issues, we upgraded the inference library on top of the PaddleX 3.0 toolkit[4], which provides extensive inference optimization and deployment features. The backward compatibility was also considered, which minimizes migration effort for users transitioning from PaddleOCR 2.x. The new inference library consists of three layers, as illustrated in Figure 10.

- **Interface Layer**: The library offers both a Python API and a CLI for user interaction. In PaddleOCR 3.0, all OCR tasks are accessed through a consistent and unified Python API. To facilitate a smooth transition for existing users, the API preserves backward compatibility for key methods and parameters. The CLI has been completely redesigned compared to PaddleOCR 2.x, introducing subcommands that clearly distinguish between different tasks and thereby providing a cleaner, more intuitive user experience.
- **Wrapper Layer**: This layer offers Pythonic wrappers for core PaddleX components, including models and pipelines. These wrappers deliver unified interfaces and flexible configuration management. In addition to maintaining the backward compatibility in the argument-based approach previously preferred in PaddleOCR 2.x, the PaddleX-style configuration file-based approach is also supported, which allows configurations to be stored and reused in a portable and reproducible manner.

[4] https://github.com/PaddlePaddle/PaddleX/tree/release/3.0

- **Foundation Layer**: At the foundation lies the PaddleX 3.0 toolkit, which forms the core of PaddleOCR 3.0. It offers powerful features for inference optimization and model deployment, which are fully integrated into PaddleOCR 3.0. Transferring the basis from PaddleOCR scripts to PaddleX ensures the separation of roles of the model training toolkit and the inference library, eliminating redundant entry points and clarifying functional boundaries. This decoupling allows each component to evolve independently, reduces user confusion, and lays the foundation for a more robust and maintainable system design.

This layered architecture ensures that higher-level components depend only on lower-level abstractions, promoting loose coupling, modularity, and ease of maintenance.

4. Deployment

An overview of the deployment capabilities of PaddleOCR 3.0 is depicted in Figure 11. To support a wide range of application scenarios, PaddleOCR 3.0 offers flexible and comprehensive deployment options, including high-performance inference, serving, and on-device deployment. In real-world production environments, OCR-related systems are often subject to constraints beyond recognition accuracy, such as latency, throughput, and hardware compatibility. PaddleOCR addresses these requirements by providing configurable deployment tools that simplify integration across various platforms. In addition, to facilitate integration with LLM applications, PaddleOCR provides an MCP server, which allows users to leverage high-performance inference pipelines or pipeline servers.

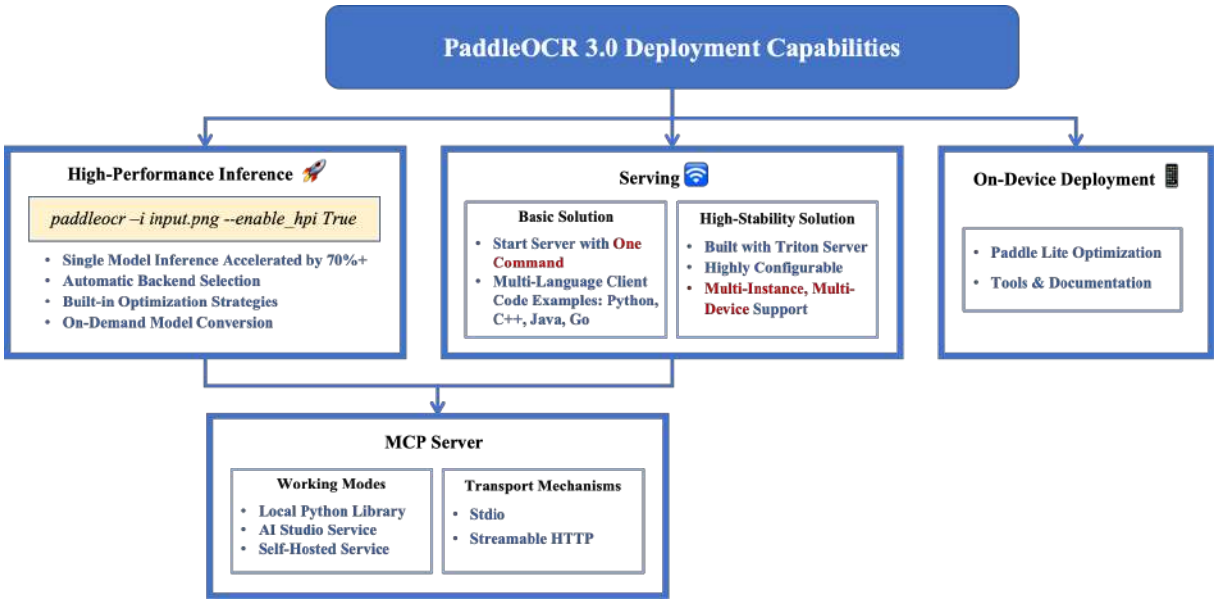


Figure 11 | Overview of the deployment capabilities of PaddleOCR 3.0. PaddleOCR 3.0 provides high-performance inference, serving, and on-device deployment capabilities. Additionally, it enables users to easily deploy an MCP server based on PaddleOCR.

4.1. High-Performance Inference

PaddleOCR 3.0 provides the high-performance inference feature that enables users to optimize runtime performance without the need to manually tune low-level configurations. High-performance inference provides notable acceleration for some key models. For instance, on

NVIDIA Tesla T4 devices, enabling high-performance inference reduces the single-model inference latency of PP-OCRv5_mobile_rec by 73.1% and that of PP-OCRv5_mobile_det by 40.4%. The key features of PaddleOCR 3.0's high-performance inference capability include:

- Automatic selection of appropriate inference backends based on the runtime environment and model characteristics, including support for Paddle Inference, OpenVINO (Intel Corporation, 2018), ONNX Runtime (Microsoft Corporation, 2018), and TensorRT (NVIDIA Corporation, 2017).
- Built-in optimization strategies such as multi-threading and FP16 inference to better utilize hardware resources.
- On-demand model conversion from PaddlePaddle static graphs to ONNX format to enable acceleration on compatible inference engines.

Users can easily achieve inference acceleration by enabling the `enable_hpi` switch, while all underlying optimization details are managed by PaddleOCR. For advanced needs, PaddleOCR also supports fine-grained tuning of high-performance inference configurations in pipelines by passing Python/command line parameters or modifying configuration files.

4.2. Serving

PaddleOCR 3.0 supports pipeline serving for building scalable and production-ready OCR-related services. Two solutions are provided:

- **Basic Serving**: A lightweight solution based on FastAPI (Ramírez, 2018) with minimal setup, suitable for rapid validation and scenarios with low concurrency requirements. For this solution, users can run any pipeline as a service with a single command via the CLI. For the client-side code, PaddleOCR 3.0 provides rich calling examples in seven programming languages: Python, C++, Java, Go, C#, Node.js, and PHP. Users can refer to these examples to quickly integrate the service capabilities into their own applications.
- **High-Stability Serving**: A more robust option built on NVIDIA Triton Inference Server (NVIDIA Corporation, 2018), which supports more advanced deployment configurations. This solution is suitable for scenarios with higher requirements for stability and performance. For example, the server can be configured to run multiple instances across multiple GPUs to fully utilize the available computing resources.

Both solutions share similar interfaces. Users can start with the basic solution for rapid validation, and then decide whether to adopt the more complex high-stability solution according to their needs, usually without significant migration costs.

4.3. On-Device Deployment

To support the deployment on resource-constrained devices, PaddleOCR 3.0 enables deployment of PP-OCR models on mobile platforms. It provides supporting tools and documentation for model optimization and integration with the Paddle-Lite[5], runtime, making it feasible to run OCR tasks efficiently on mobile devices.

[5] https://github.com/PaddlePaddle/Paddle-Lite

4.4. MCP Server

PaddleOCR 3.0 provides a lightweight MCP server, enabling smooth integration of PaddleOCR's core capabilities into any MCP-compatible host. Both the OCR and PP-StructureV3 pipelines are currently accessible as tools via the MCP server.

Built on top of the PaddleOCR inference library, the MCP server supports various inference and deployment methods provided by PaddleOCR. At present, it can operate in one of three working modes:

- **Local:** Runs the PaddleOCR pipeline directly on the local machine using the installed Python library. This mode is suitable for offline usage and situations with strict data privacy requirements. High-performance inference can be activated to accelerate the inference process.
- **AI Studio:** Utilizes cloud services hosted by the PaddlePaddle AI Studio community (PaddlePaddle Team, 2019). This mode is ideal for quickly trying out features, validating solutions, and for no-code development scenarios.
- **Self-Hosted:** Connects to a user-hosted PaddleOCR service. This mode offers the advantages of pipeline serving and high flexibility, making it well-suited for scenarios requiring custom service configurations.

Regardless of the selected working mode, setting up the PaddleOCR MCP server is straightforward for users. Example configuration files are included in the appendix 5 for reference. Additionally, the PaddleOCR MCP server supports both stdio and Streamable HTTP transport mechanisms, offering flexibility for a wide range of deployment scenarios. With its adaptable architecture and support for multiple deployment modes, the PaddleOCR MCP server can effectively address diverse real-world application needs, delivering robust and scalable solutions for both individual developers and enterprise users.

5. Conclusion

PaddleOCR has been dedicated to the field of OCR and document parsing for many years, aiming to provide more valuable technical solutions. PaddleOCR 3.0 is a milestone upgrade, with technologies like PP-OCRv5, PP-StructureV3, and PP-ChatOCRv4 set to play a significant role in the era of large-scale models. Moving forward, we will continue to expand our models, including the upcoming release of multilingual text recognition models, multimodal OCR, and document parsing models. If you find PaddleOCR 3.0 useful or wish to use it in your projects, please kindly cite this technical report.

References

J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

R. AI. Rolmocr: A faster, lighter open source ocr model, 2025.

Baidu-ERNIE-Team. Ernie 4.5 technical report, 2025.

L. Blecher, G. Cucurull, T. Scialom, and R. Stojnic. Nougat: Neural optical understanding for academic documents, 2023.

breezedeus. Pix2text. https://github.com/breezedeus/Pix2Text, 2022. Accessed: 2025-06-23.

R. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(7):690–706, 1996. doi: 10.1109/34.506792.

Z. Chen, W. Wang, Y. Cao, Y. Liu, Z. Gao, E. Cui, J. Zhu, S. Ye, H. Tian, Z. Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. arXiv preprint arXiv:2412.05271, 2024.

C. Cui, T. Gao, S. Wei, Y. Du, R. Guo, S. Dong, B. Lu, Y. Zhou, X. Lv, Q. Liu, X. Hu, D. Yu, and Y. Ma. Pp-lcnet: A lightweight cpu convolutional neural network, 2021. URL https://arxiv.org/abs/2109.15099.

Docling Team. Docling. https://github.com/docling-project/docling, 2024. Accessed: 2025-06-23.

Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang, et al. Pp-ocr: A practical ultra lightweight ocr system. arXiv preprint arXiv:2009.09941, 2020.

Y. Du, C. Li, R. Guo, C. Cui, W. Liu, J. Zhou, B. Lu, Y. Yang, Q. Liu, X. Hu, et al. Pp-ocrv2: Bag of tricks for ultra lightweight ocr system. arXiv preprint arXiv:2109.03144, 2021.

Y. Du, Z. Chen, C. Jia, X. Yin, T. Zheng, C. Li, Y. Du, and Y.-G. Jiang. Svtr: Scene text recognition with a single visual model. arXiv preprint arXiv:2205.00159, 2022.

Filimoa. open-parse. https://github.com/Filimoa/open-parse, 2024. Accessed: 2025-06-23.

I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks, 2014. URL https://arxiv.org/abs/1312.6082.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

J. Ha, R. M. Haralick, and I. T. Phillips. Recursive xy cut using bounding boxes of connected components. In Proceedings of 3rd International Conference on Document Analysis and Recognition, volume 2, pages 952–955. IEEE, 1995.

hiroi sora. Umi-ocr. https://github.com/hiroi-sora/Umi-OCR, 2022. Accessed: 2025-06-23.

W. Hu, X. Cai, J. Hou, S. Yi, and Z. Lin. Gtc: Guided training of ctc towards efficient and accurate scene text recognition. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 11005–11012, 2020.

Intel Corporation. OpenVINO Toolkit. https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html, 2018. Accessed: 2025-06-23.

KevinHuSh. ragflow. https://github.com/infiniflow/ragflow, 2023. Accessed: 2025-06-23.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33:9459–9474, 2020.

C. Li, R. Guo, J. Zhou, M. An, Y. Du, L. Zhu, Y. Liu, X. Hu, and D. Yu. Pp-structurev2: A stronger document analysis system. arXiv preprint arXiv:2210.05391, 2022a.

C. Li, W. Liu, R. Guo, X. Yin, K. Jiang, Y. Du, Y. Du, L. Zhu, B. Lai, X. Hu, et al. Pp-ocrv3: More attempts for the improvement of ultra lightweight ocr system. arXiv preprint arXiv:2206.03001, 2022b.

C. Li, W. Liu, R. Guo, X. Yin, K. Jiang, Y. Du, Y. Du, L. Zhu, B. Lai, X. Hu, et al. Pp-ocrv3: More attempts for the improvement of ultra lightweight ocr system. arXiv preprint arXiv:2206.03001, 2022c.

H. Liu, C. Cui, Y. Du, Y. Liu, and G. Pan. Pp-formulanet: Bridging accuracy and efficiency in advanced formula recognition. arXiv preprint arXiv:2503.18382, 2025.

Y. Ma, D. Yu, T. Wu, and H. Wang. Paddlepaddle: An open-source deep learning platform from industrial practice. Frontiers of Data and Domputing, 1(1):105–115, 2019.

Microsoft Corporation. ONNX Runtime. https://github.com/microsoft/onnxruntime, 2018. Accessed: 2025-06-23.

S. Mori, H. Nishida, and H. Yamada. Optical Character Recognition. John Wiley & Sons, 1999.

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

F. Ni, K. Huang, Y. Lu, W. Lv, G. Wang, Z. Chen, and Y. Liu. Pp-docbee: Improving multimodal document understanding through a bag of tricks. arXiv preprint arXiv:2503.04065, 2025.

NVIDIA Corporation. TensorRT. https://developer.nvidia.com/tensorrt, 2017. Accessed: 2025-06-23.

NVIDIA Corporation. Triton Inference Server. https://github.com/triton-inference-server/server, 2018. Accessed: 2025-06-23.

L. Ouyang, Y. Qu, H. Zhou, J. Zhu, R. Zhang, Q. Lin, B. Wang, Z. Zhao, M. Jiang, X. Zhao, et al. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 24838–24848, 2025.

PaddlePaddle Team. Ai studio. https://aistudio.baidu.com, 2019. Accessed: 2025-06-23.

V. Paruchuri. Marker. https://github.com/VikParuchuri/marker, 2023. Accessed: 2025-06-23.

J. Poznanski, J. Borchardt, J. Dunkelberger, R. Huff, D. Lin, A. Rangapur, C. Wilhelm, K. Lo, and L. Soldaini. olmocr: Unlocking trillions of tokens in pdfs with vision language models. arXiv preprint arXiv:2502.18443, 2025.

S. Ramírez. FastAPI. https://github.com/fastapi/fastapi, 2018. Accessed: 2025-06-23.

B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, 2015. URL https://arxiv.org/abs/1507.05717.

T. Sun, C. Cui, Y. Du, and Y. Liu. Pp-doclayout: A unified document layout detection model to accelerate large-scale data construction. arXiv preprint arXiv:2503.17213, 2025.

Unstructured-IO. unstructured. https://github.com/Unstructured-IO/unstructured, 2022. Accessed: 2025-06-23.

F. Verhoeven, T. Magne, and O. Sorkine-Hornung. Uvdoc: neural grid-based document unwarping. In SIGGRAPH Asia 2023 Conference Papers, pages 1–11, 2023.

B. Wang, C. Xu, X. Zhao, L. Ouyang, F. Wu, Z. Zhao, R. Xu, K. Liu, Y. Qu, F. Shang, et al. Mineru: An open-source solution for precise document content extraction. arXiv preprint arXiv:2409.18839, 2024.

H. Wei, C. Liu, J. Chen, J. Wang, L. Kong, Y. Xu, Z. Ge, L. Zhao, J. Sun, Y. Peng, et al. General ocr theory: Towards ocr-2.0 via a unified end-to-end model. 2024.

A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024.

A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Appendix

A. Acknowledgments

We gratefully acknowledge all individuals who supported this work through their invaluable contributions to data construction, deployment, testing, project maintenance, product development, online demo creation, and operations. Their dedication and efforts have played a crucial role in the successful advancement and ongoing improvement of this project.

Baoku Yu	Jiahua Wang	Siyu Cheng	Yiqiao Zhou
Chang Xu	Jianying Qu	Suyin Liang	Ye Han
Chao Han	Jiaxin Sui	Tao Luo	Yongkun Du
Chunli Xie	Jinghui Duan	Tianyu Zheng	Zewu Wu
Guanzhong Wang	JingsongLiu	Xiaolong Ma	Zeyu Luo
Haitao Yu	Mengmeng Guo	Xin Li	Zhe Wang
Hengxin Chen	Min Zhuang	Xin Wang	Zhongkai Sun
Hong Cheng	Runlong Li	Xinran Liu	
Jiahao Bai	Shengjian Guo	Yimin Gao	

We would also like to acknowledge the invaluable contributions of open-source developers on GitHub, including but not limited to @timminator, @ackinc, @Appla, @co63oc, @jk4e, and many others whose work has inspired and supported our project.

Finally, we express our sincere gratitude to all project contributors for their long-term support and valuable input. Their efforts have greatly advanced the development and improvement of this project. Figure 12 shows the avatars of these contributors, as collected from the PaddleOCR GitHub repository.

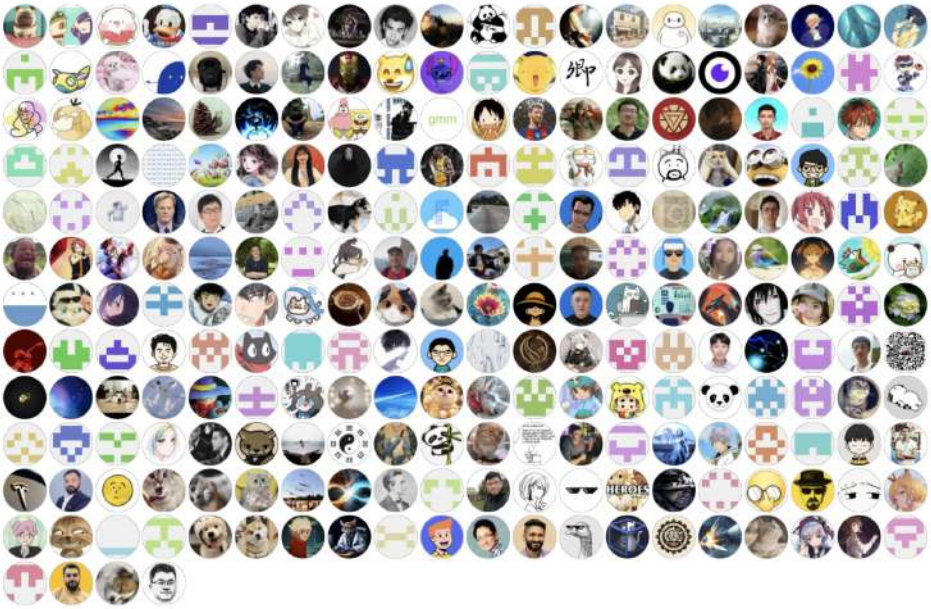


Figure 12 | Avatars of long-term contributors to the PaddleOCR project.

B. Usage of command and API details

To use PaddleOCR 3.0, you can simply install the paddleocr package from PyPI. PaddleOCR 3.0 provides command-line interface (CLI) and Python API for users to use conveniently.

B.1. Run inference by CLI

We provide convenient CLI methods for users to quickly experience the capabilities of PP-OCRv5, PP-StructureV3, and PP-ChatOCRv4, as follows:

```
# Run PP-OCRv5 inference
paddleocr ocr -i test.png \
        --use_doc_orientation_classify False \
        --use_doc_unwarping False \
        --use_textline_orientation False

# Run PP-StructureV3 inference
paddleocr pp_structurev3 -i test.png \
        --use_doc_orientation_classify False \
        --use_doc_unwarping False

# Get the Qianfan API Key at first, then run PP-ChatOCRv4
paddleocr pp_chatocrv4_doc -i test.png \
        -k number \
        --qianfan_api_key your_api_key \
        --use_doc_orientation_classify False \
        --use_doc_unwarping False
```

B.2. Run inference by Python API

We also provide a clean interface to facilitate users in using and integrating it into their own projects.

1. PP-OCRv5 Example

```
# Initialize PaddleOCR instance
from paddleocr import PaddleOCR
ocr = PaddleOCR(
    use_doc_orientation_classify=False,
    use_doc_unwarping=False,
    use_textline_orientation=False)

# Run OCR inference on a sample image
result = ocr.predict(input="test.png")

# Visualize the results and save the JSON results
for res in result:
    res.print()
    res.save_to_img("output")
    res.save_to_json("output")
```

2. PP-StructureV3 Example

```
# Initialize PPStructureV3 instance
from paddleocr import PPStructureV3

pipeline = PPStructureV3(
    use_doc_orientation_classify=False,
    use_doc_unwarping=False)

# Run PPStructureV3 inference
output = pipeline.predict(input="test.png")
```

```

# Visualize the results and save the JSON results
for res in output:
    res.print()
    res.save_to_json(save_path="output")
    res.save_to_markdown(save_path="output")
```

3. PP-ChatOCRv4 Example

```
from paddleocr import PPChatOCRv4Doc

chat_bot_config = {
    "module_name": "chat_bot",
    "model_name": "xxx",
    "base_url": "https://qianfan.baidubce.com/v2",
    "api_type": "openai",
    "api_key": "api_key",  # your api_key
}

retriever_config = {
    "module_name": "retriever",
    "model_name": "embedding-v1",
    "base_url": "https://qianfan.baidubce.com/v2",
    "api_type": "qianfan",
    "api_key": "api_key",  # your api_key
}

# Initialize PPChatOCRv4Doc instance
pipeline = PPChatOCRv4Doc(
    use_doc_orientation_classify=False,
    use_doc_unwarping=False)

visual_predict_res = pipeline.visual_predict(
    input="test.png",
    use_common_ocr=True,
    use_seal_recognition=True,
    use_table_recognition=True)

mllm_predict_info = None
use_mllm = False
visual_info_list = []

for res in visual_predict_res:
    visual_info_list.append(res["visual_info"])
    layout_parsing_result = res["layout_parsing_result"]

vector_info = pipeline.build_vector(
    visual_info_list,
    flag_save_bytes_vector=True,
    retriever_config=retriever_config)

chat_result = pipeline.chat(
    key_list=["number of people"],
    visual_info=visual_info_list,
    vector_info=vector_info,
    mllm_predict_info=mllm_predict_info,
    chat_bot_config=chat_bot_config,
    retriever_config=retriever_config)
print(chat_result)
```

C. More details on MCP host configuration

Here are several example configurations for the MCP host in Claude for Desktop, illustrating how to connect to a PaddleOCR MCP server. Each configurable parameter may be specified either via environment variables (as shown in the `env` field) or via command-line arguments (as shown in the `args` field).

1. Using the local Python library:

```
{
  "mcpServers": {
    "paddleocr-ocr": {
      "command": "paddleocr_mcp",
      "args": [
        "--device", "gpu:1"
      ],
      "env": {
        "PADDLEOCR_MCP_PIPELINE": "OCR",
        "PADDLEOCR_MCP_PPOCR_SOURCE": "local"
      }
    }
  }
}
```

2. Using a PaddlePaddle AI Studio service:

```
{
  "mcpServers": {
    "paddleocr-ocr": {
      "command": "paddleocr_mcp",
      "args": [
        "--timeout", "60"
      ],
      "env": {
        "PADDLEOCR_MCP_PIPELINE": "OCR",
        "PADDLEOCR_MCP_PPOCR_SOURCE": "aistudio",
        "PADDLEOCR_MCP_SERVER_URL": "<server-url>",
        "PADDLEOCR_MCP_AISTUDIO_ACCESS_TOKEN": "<access-token>"
      }
    }
  }
}
```

3. Using a self-hosted service:

```
{
  "mcpServers": {
    "paddleocr-ocr": {
      "command": "paddleocr_mcp",
      "args": [],
      "env": {
        "PADDLEOCR_MCP_PIPELINE": "OCR",
        "PADDLEOCR_MCP_PPOCR_SOURCE": "self_hosted",
        "PADDLEOCR_MCP_SERVER_URL": "<server-url>"
      }
    }
  }
}
```