A Scalable Microservices Architecture for AI-Driven Document Layout Analysis and Content Digitization

Franco Calle, **Renata Calle**
Independent Researchers
Email: francocalle93@gmail.com, renatacallef@gmail.com

Abstract

Document digitization remains a critical challenge in information processing, particularly for complex layouts containing mixed content types such as text, tables, figures, and mathematical formulas. This paper presents a novel microservices-based architecture that combines state-of-the-art AI models for document layout analysis, reading order prediction, and content extraction. Our proposed system integrates YOLO-based layout detection, Microsoft's LayoutReader for reading order prediction, and multi-provider large language models (LLMs) for content digitization. The architecture design features ultra-isolation session management with UUID fingerprinting, two-tier caching mechanisms, and comprehensive performance optimization strategies. We present the theoretical framework and architectural design for processing multiple document formats including PDF, PPTX, DOC, DOCX, XLSX, and image formats, with a focus on scalability, reliability, and modularity in the system design.

Keywords: document analysis, layout detection, microservices, artificial intelligence, content extraction, reading order prediction, YOLO, LayoutLMv3, multi-LLM integration

1 Introduction

The exponential growth of digital documents across industries has created unprecedented demand for automated document processing systems capable of understanding complex layouts and extracting structured content. Traditional Optical Character Recognition (OCR) systems, while effective for plain text extraction, fall short when dealing with modern documents containing mixed content types such as tables, figures, mathematical formulas, and multi-column layouts [1]. The challenge extends beyond simple text recognition to encompass document understanding, reading order prediction, and semantic content extraction.

Contemporary document processing faces several critical challenges: (1) **Layout Complexity**: Modern documents exhibit sophisticated layouts with nested structures, overlapping elements, and varied content types requiring spatial reasoning beyond traditional OCR capabilities. (2) **Reading Order Prediction**: Determining the correct sequence for consuming document content, particularly in multi-column layouts and documents with complex spatial arrangements [5]. (3) **Content Extraction Accuracy**: Preserving semantic relationships between document elements while maintaining structural integrity during digitization. (4) **Scalability**: Processing large volumes of documents with varying formats and layouts while maintaining consistent quality and performance.

Recent advances in artificial intelligence, particularly in computer vision and large language models (LLMs), have opened new possibilities for addressing these challenges. State-of-the-art approaches leverage deep learning models for layout detection [7], transformer-based architectures for reading order prediction [6], and LLMs for intelligent content extraction [4]. However, integrating these technologies into a cohesive, production-ready system that maintains high performance, reliability, and scalability remains a significant engineering challenge.

This paper presents a comprehensive microservices architecture that addresses these challenges through novel integration of state-of-the-art AI models and robust engineering practices. Our proposed system design combines YOLO-based layout detection with configurable confidence and IoU thresholds, Microsoft's LayoutReader model for reading order prediction, and multi-provider LLM integration supporting 5 major providers with dynamic model selection and automatic failover mechanisms.

Key Contributions: (1) A scalable microservices architecture seamlessly integrating multiple state-of-the-art AI models with asynchronous processing capabilities. (2) Novel ultra-isolation session management with UUID fingerprinting preventing cross-contamination in multi-tenant environments. (3) Mathematical formulations for layout detection, reading order prediction, and multi-LLM selection algorithms. (4) Comprehensive two-tier caching system design with L1 memory and L2 disk persistence. (5) Detailed architectural patterns and design considerations for building production-ready document processing systems.

2 Related Work

2.1 Document Layout Analysis

Document layout analysis has evolved from rule-based approaches to deep learning methodologies. Early systems relied on connected component analysis and geometric heuristics [8]. The introduction of convolutional neural networks revolutionized the field, with frameworks like Faster R-CNN and YOLO being adapted for document element detection [7].

Recent transformer-based architectures have further improved layout understanding. LayoutLM and its variants [2] demonstrate effectiveness in combining textual and spatial information through joint pre-training on large-scale document datasets. DocBank [3] and PubLayNet [8] provide standardized benchmarks enabling systematic comparison, with modern systems achieving over 90% accuracy in layout element classification.

2.2 Reading Order Prediction

Reading order prediction addresses the fundamental challenge of determining logical sequence for consuming document content. Traditional approaches relied on geometric heuristics such as top-to-bottom, left-to-right scanning, proving inadequate for complex multi-column layouts [5].

Microsoft's LayoutReader [6] represents significant advancement, introducing transformer-based approach achieving state-of-the-art performance. The model, pre-trained on ReadingBank dataset containing over 500,000 document pages, demonstrates effectiveness of learning spatial relationships through self-supervised learning, achieving approximately 97% BLEU score.

2.3 Large Language Model Integration

The emergence of large language models has transformed content extraction capabilities [4]. Modern approaches leverage vision-language models processing visual layout and textual content simultaneously, enabling accurate extraction of complex elements. Multi-provider architectures provide reliability through redundancy and dynamic model selection based on content complexity.

3 System Architecture

Our document detection system implements a microservices architecture designed for scalability, maintainability, and high performance. The system comprises five core processing services orchestrated through Docker Compose, with Redis for state management and RabbitMQ for asynchronous message passing.

3.1 Mathematical Framework

Let $D = \{d_1, d_2, \ldots, d_n\}$ represent a set of input documents, where each document d_i contains a sequence of pages $P_i = \{p_1, p_2, \ldots, p_{|P_i|}\}$. For each page p_j, we define the processing pipeline as:

$$\Phi(p_j) = C \circ R \circ L \circ O \circ I(p_j) \tag{1}$$

where:

- I: Ingestion function with format detection and preprocessing
- O: Orientation detection function (currently disabled for performance)
- L: Layout detection function using YOLO or LayoutLMv3
- R: Reading order prediction using LayoutReader
- C: Content digitization and compilation using multi-LLM providers

3.2 Core Microservices

3.2.1 Multi-Format Document Ingestion Service

The ingestion service serves as the system's entry point, supporting 8+ document formats: PDF, PNG, JPG, JPEG, PPTX, DOC, DOCX, XLSX. The service implements adaptive preprocessing optimizing images to 1–3MB per page while maintaining quality sufficient for downstream processing.

Image Optimization Algorithm: Given input image I with dimensions (w, h) and target size $S_{target} \in [1MB, 3MB]$, the optimization follows these steps:

1. Calculate adaptive DPI: $dpi = \text{adaptiveDpi}(w, h)$ in range [150, 300]
2. Render image: $I' = \text{render}(I, dpi)$
3. Check size: if $|I'| \leq S_{target}$, return I'
4. Binary search quality: $q = \text{binarySearchQuality}(I', S_{target})$ in [60%, 95%]
5. Compress: $I'' = \text{compress}(I', q)$
6. If still oversized: $I'' = \text{resizeProportional}(I'', S_{target})$

3.2.2 Layout Detection Service

The layout detection service implements three distinct models with mathematical formulations:

YOLO-based Detection: Using doclayout_yolo_ft.pt with confidence threshold $\theta_c = 0.25$ and IoU threshold $\theta_{IoU} = 0.45$. For detected bounding box $b_i = (x_1, y_1, x_2, y_2)$ with confidence c_i, we filter detections:

$$B_{filtered} = \{b_i \mid c_i \geq \theta_c\} \tag{2}$$

Bounding Box Merging: To consolidate overlapping detections using IoU threshold $\theta_{IoU} = 0.45$:

$$IoU(b_i, b_j) = \frac{Area(b_i \cap b_j)}{Area(b_i \cup b_j)} \tag{3}$$

The merging algorithm groups detections with IoU $\geq \theta_{IoU}$ and consolidates them using depth-first search grouping strategy.

3.2.3 Reading Order Prediction Service

This service integrates Microsoft's LayoutReader model with approximately 360M parameters. Given a set of detected components $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ with bounding boxes $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$, the reading order prediction finds optimal permutation:

$$\pi^* = \arg\max_{\pi \in S_n} P(\pi \mid \mathcal{B}, \mathcal{T}) \tag{4}$$

where π is a permutation of $\{1, 2, \ldots, n\}$, S_n is the set of all permutations, and $\mathcal{T}$ represents text features.

Processing Details: Coordinates are normalized to range [0, 1000] and processed with MAX_LEN=510 tokens. The model processes spatial relationships through transformer attention mechanisms designed for high BLEU score performance and efficient GPU throughput.

3.2.4 Multi-Provider Digitization Service

The digitization service implements AI-powered content extraction using 5 LLM providers: OpenAI (GPT-4), Anthropic (Claude), Groq (Llama), Google (Gemini), Together AI (Qwen). The service features dynamic model selection based on component complexity analysis.

Given component c with complexity score $\rho(c)$ and available providers $\mathcal{P} = \{p_1, p_2, \ldots, p_k\}$, provider selection follows:

$$p^* = \arg\max_{p \in \mathcal{P}} \frac{\text{Benefits}(p)}{\text{Costs}(p)} \tag{5}$$

$$\text{where Benefits}(p) = \alpha \cdot accuracy(p) + \beta \cdot availability(p)$$

$$\text{Costs}(p) = \gamma \cdot cost(p) + \delta \cdot latency(p)$$

where $\alpha, \beta, \gamma, \delta$ are weighting parameters based on component complexity $\rho(c)$.

Ultra-Isolation Session Management: Each processing session receives UUID fingerprinting with complete isolation:

$$\text{Session}_i = \{UUID_i, Resources_i, Cache_i, State_i\} \tag{6}$$

3.3 Performance Optimization Framework

3.3.1 Two-Tier Caching System

The caching architecture implements L1 memory cache with LRU eviction and L2 disk persistence:

$$P_{hit} = P(L1_{hit}) + P(L1_{miss}) \cdot P(L2_{hit}) \tag{7}$$

where high cache hit rates are achievable through proper cache sizing and eviction policies.

Composite Cache Keys:

$$Key = H(job\_id \oplus component\_id \oplus session\_id \oplus model\_version) \tag{8}$$

where H is a cryptographic hash function and $\oplus$ denotes concatenation.

4 Performance Optimization Framework

4.1 Optimization Strategies

The proposed architecture incorporates systematic performance optimization strategies designed to maximize throughput:

$$\text{Throughput}_{opt} = \alpha \cdot \text{Parallelism} + \beta \cdot \text{Caching} \tag{9}$$

$$\text{Latency}_{opt} = \frac{\text{Latency}_{base}}{\text{Optimization Factor}} \tag{10}$$

$$\text{Memory}_{opt} = \text{Memory}_{base} - \text{Optimization Savings} \tag{11}$$

4.2 Model Selection Framework

Table 1: Layout Detection Model Characteristics

Model	Accuracy Focus	Speed Profile	Memory Usage	GPU Req'd
YOLO	Balanced	Fast	Low	No
LayoutLMv3	High	Moderate	High	Yes
Agentic LLM	Flexible	Variable	Medium	No

4.3 Multi-LLM Provider Considerations

4.4 End-to-End System Design

Processing Pipeline Model: The system architecture targets optimal processing through:

$$T_{total} = T_{ing} + T_{lay} + T_{read} + T_{dig} + T_{comp} \tag{12}$$

where each component can be optimized independently for overall system improvement.

Scalability Design Goals:

- End-to-End: Target high throughput with complexity-aware processing

Table 2: LLM Provider Characteristics

Provider	Strength	Speed	Cost	Avail	Limits
OpenAI GPT-4	Accuracy	Moderate	High	High	Standard
Claude 3 Sonnet	Balance	Good	Medium	High	Moderate
Gemini Pro	Speed	Fast	Low	High	High
Llama 3.1 (Groq)	Cost	V. Fast	V. Low	Medium	Limited
Qwen 2.5-VL	Vision	Good	Low	Medium	Moderate

- Reading Order: Leverage GPU acceleration when available
- Layout Detection: Support multiple models with varying speed/accuracy tradeoffs
- Cache Performance: Design for high hit rates through intelligent key management
- Service Availability: Multi-provider redundancy for high availability

5 Implementation and Quality Assurance

5.1 Async Message Processing

The system uses aio_pika for RabbitMQ integration with optimized connection pooling. The processing pipeline follows:

1. Initialize connection pool with N workers
2. For each incoming message: assign to available worker
3. Create isolated session with UUID fingerprinting
4. Process component through pipeline
5. Update job status and publish results

5.2 Quality Assurance Framework

Component Validation: Each extracted component undergoes validation:

$$Valid(c) = \begin{cases} True & \text{if } Area(c) > \theta_{area} \\ & \wedge Confidence(c) > \theta_{conf} \\ False & \text{otherwise} \end{cases} \tag{13}$$

where θ_{area} and θ_{conf} are configurable thresholds for area and confidence respectively.

Contamination Detection: Session isolation validation ensures no cross-contamination through fingerprint comparison and resource tracking. Detected contamination triggers automatic session reset and re-processing.

6 Conclusion and Future Work

This paper presented a comprehensive microservices architecture for AI-driven document processing through novel integration of multiple AI models and advanced engineering practices. The proposed system design offers a scalable framework for building production-ready document understanding systems.

Key Technical Achievements:

- Mathematical formulation of layout detection and reading order prediction
- Ultra-isolation session management preventing cross-contamination
- Dynamic multi-LLM selection with automatic failover
- Two-tier caching architecture with comprehensive performance optimization
- Scalable async processing with horizontal scaling capabilities

Future Research Directions:

- Integration of multimodal foundation models (GPT-4V, Gemini Vision)
- Advanced reading order algorithms using graph neural networks
- Real-time processing optimization with edge computing deployment
- Federated learning for privacy-preserving model improvement
- Extension to 3D document processing and augmented reality applications

The system's production deployment demonstrates practical applicability across diverse industries while maintaining research-grade accuracy and performance standards suitable for academic and commercial applications.

Acknowledgment

The authors thank the open-source community for tools enabling this research, including LayoutReader, LayoutLMv3, YOLO developers, and LLM providers facilitating multi-provider integration. Special recognition to early system users providing valuable feedback during development and optimization phases.

References

[1] Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. Document ai: Benchmarks, models and applications. *arXiv preprint arXiv:2111.08609*, 2021. Comprehensive survey on Document AI covering benchmarks, models, and applications.

[2] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document

ai with unified text and image masking. pages 4847–4857, 2022.

[3] Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. Docbank: A benchmark dataset for document layout analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 949–960. International Committee on Computational Linguistics, 2020.

[4] Dongsheng Wang, Zhiqiang Xu, Wei Ma, Xiang Lu, Bo Li, and Zijian Chen. Docllm: A layout-aware generative language model for multimodal document understanding. *arXiv preprint arXiv:2401.00908*, 2024. Layout-aware generative language model for multimodal document understanding.

[5] Zilong Wang, Yiheng Xu, Lei Cui, Jingbo Shang, and Furu Wei. Layoutreader: Pre-training of text and layout for reading order detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4735–4744. Association for Computational Linguistics, 2021. Achieves 97% BLEU score on ReadingBank dataset with 500K document pages.

[6] Zilong Wang, Yiheng Xu, Lei Cui, Jingbo Shang, and Furu Wei. Layoutreader: Pre-training of text and layout for reading order detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4735–4744. Association for Computational Linguistics, 2021.

[7] Zhiyuan Zhao, Hengrui Kang, Bin Wang, and Conghui He. Doclayout-yolo: Enhancing document layout analysis through diverse synthetic data and global-to-local adaptive perception. *arXiv preprint arXiv:2410.12628*, 2024. State-of-the-art YOLO-based approach for document layout analysis achieving 70.3% mAP on D4LA dataset.

[8] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE, 2019. Best paper award winner at ICDAR 2019.