Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

TongGuOCR: A Layout-Aware and Token-Augmented OCR Framework for Chinese Historical Documents

Zhongheng Zhou
South China University of Technology
Yi Sun
South China University of Technology
Huiguo He
South China University of Technology
Yuyi Zhang
South China University of Technology
Peirong Zhang
South China University of Technology
Yulin Fang
South China University of Technology
Dezhi Peng
Huawei Technologies (China)
Minghui Liao
Huawei Technologies (China)
Lianwen Jin
lianwen.jin@gmail.com
South China University of Technology

Article

Keywords:

Posted Date: August 7th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10564956/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

TongGuOCR: A Layout-Aware and Token-Augmented OCR Framework for Chinese Historical Documents

Zhongheng Zhou[1†], Yi Sun[1†], Huiguo He[1*†], Yuyi Zhang[1], Peirong Zhang[1], Yulin Fang[1], Dezhi Peng[2], Minghui Liao[2], Lianwen Jin[1*]

[1]School of Electronic and Information Engineering, South China University of Technology, Guangzhou, 510641, China.
[2]Huawei Technologies Co., Ltd., Dongguan, 511700, China.

*Corresponding author(s). E-mail(s): hehuiguo@scut.edu.cn; lianwen.jin@gmail.com;
Contributing authors: zzh21249@gmail.com; siryhahaha2035@gmail.com; yuyi.zhang11@foxmail.com; eeprzhang@mail.scut.edu.cn; 17846944407@163.com; pengdzscut@foxmail.com; mhliao@foxmail.com;
†These authors contributed equally to this work.

Abstract

Chinese historical documents preserve valuable cultural heritage, but many collections remain accessible only as scanned page images, preventing full-text retrieval, collation, and computational analysis. Optical character recognition (OCR) can bridge this gap, but accurate transcription remains challenging because historical documents often contain complex layouts, rare characters, and nontrivial reading orders. We propose **TongGuOCR**, a layout-aware and token-augmented OCR framework for Chinese historical documents. First, a Layout-Aware Preprocessing module constructs and refines locally coherent recognition blocks to preserve local context while reducing interference across regions. Second, a Token-Augmented Recognition module augments the transcription target at two complementary levels: character-level vocabulary expansion gives each rare glyph a direct one-token representation and shortens its decoding path, while line-to-line transition modeling injects discrete spatial displacement tokens that guide the decoder along complex reading paths without requiring precise coordinates. Experiments on two Chinese historical document OCR benchmarks show that TongGuOCR outperforms representative traditional task-specific OCR models, general-purpose multimodal large language models (MLLMs), and OCR-oriented MLLMs. On the more challenging M^{5}HisDoc benchmark, TongGuOCR achieves 93.76 AR and reduces NED from 10.43 to 6.15 and RO-ED from 7.53 to 3.49 relative to the best competing score for each metric. The model weights and inference code are available at https://github.com/jzzh2004/TongGuOCR.

1 Introduction

Chinese historical documents preserve textual records spanning thousands of years and constitute a vital part of East Asian written cultural heritage [1–6]. To preserve and broaden access to these collections, large-scale digitization programs have converted many of them into scanned page images [7]. However, page images alone cannot support full-text retrieval, textual collation, or the computational analysis increasingly required in digital humanities and heritage science [8]. Optical

character recognition (OCR) bridges this gap by converting scanned pages into machine-readable text, enabling cross-document collation, entity extraction [9], ancient-to-modern translation [10], knowledge graph construction [11], and large-scale computational analysis of classical texts [12]. OCR therefore serves as a foundational enabling technology for the scholarly and computational use of digitized historical document collections [13].

Earlier OCR studies on Chinese historical documents mostly adopted multi-stage pipelines that combine layout analysis, text-line or character detection, recognition, and reading-order recovery [13, 17–19]. These methods provide explicit structural modeling and fine-grained control over page decomposition and recognition. More recently, general-purpose multimodal large language models (MLLMs) and OCR-oriented MLLMs have substantially advanced document OCR through large-scale visual-language pretraining, direct image-to-text generation, and integrated layout analysis, reading-order prediction, and recognition [15, 16, 20–23].

However, these methods still face significant challenges when applied to Chinese historical documents. First, page layouts are intricate and diverse, often mixing vertical columns, interlinear annotations, and irregular text blocks. Neither end-to-end OCR models nor traditional pipelines fully address this challenge. End-to-end models typically take full-page images as input, but resizing can reduce image resolution, and unrelated regions can interfere with recognition. Traditional pipelines instead detect and recognize individual text lines, preserving character detail but losing the local context needed for coherent transcription. A suitable intermediate recognition granularity is therefore needed to balance character visibility and contextual coverage.

Second, Chinese historical documents contain a large and long-tailed inventory of rare characters. Traditional classification-based recognizers rely on a predefined character set, leaving characters outside the label space unsupported and resulting in poor recognition of rare categories. MLLM-based recognizers face a different limitation: tokenizers trained primarily on modern text often decompose rare characters into multiple byte-level or subword fragments. This fragmentation weakens the correspondence between a visual glyph and its output representation, requiring multiple decoding steps for a single character.

Third, complex reading orders make line-to-line transitions difficult during transcription, even after layout analysis and region cropping. Multiple vertical columns, interlinear annotations, and marginal text can interrupt the regular reading flow, leaving the decoder uncertain about where to continue after completing a line. Without explicit spatial guidance, this ambiguity may lead to skipped or repeated lines and cross-line confusion. Supervision of the spatial displacement between successive lines is therefore needed to help the decoder follow the intended reading order. These challenges often occur simultaneously, leading to coupled errors in character recognition and reading-order recovery, as illustrated in Figure 1.

To address these challenges, we propose **TongGuOCR**, a layout-aware and token-augmented OCR framework consisting of two modules: Layout-Aware Preprocessing and Token-Augmented Recognition. Specifically, **Layout-Aware Preprocessing** addresses the recognition-granularity problem by grouping consecutive text lines into locally coherent blocks and refining them into OCR-friendly crops. This process preserves local context while reducing interference from unrelated regions. **Token-Augmented Recognition** addresses the remaining two challenges through two complementary designs. Character-level vocabulary expansion gives rare characters single-token representations and thereby removes the fragmentation introduced by a tokenizer trained on modern text, while line-to-line transition modeling turns the implicit reading order into explicit supervision at each line boundary. A three-stage supervised fine-tuning (SFT) strategy progressively adapts the recognition model to historical documents and aligns its training inputs with the recognition blocks used during inference.

The main contributions of this work are as follows:

- We propose **TongGuOCR**, a layout-aware and token-augmented OCR framework for Chinese historical documents, consisting of two modules: Layout-Aware Preprocessing and Token-Augmented Recognition.
- We introduce a **Layout-Aware Preprocessing** module that constructs locally coherent

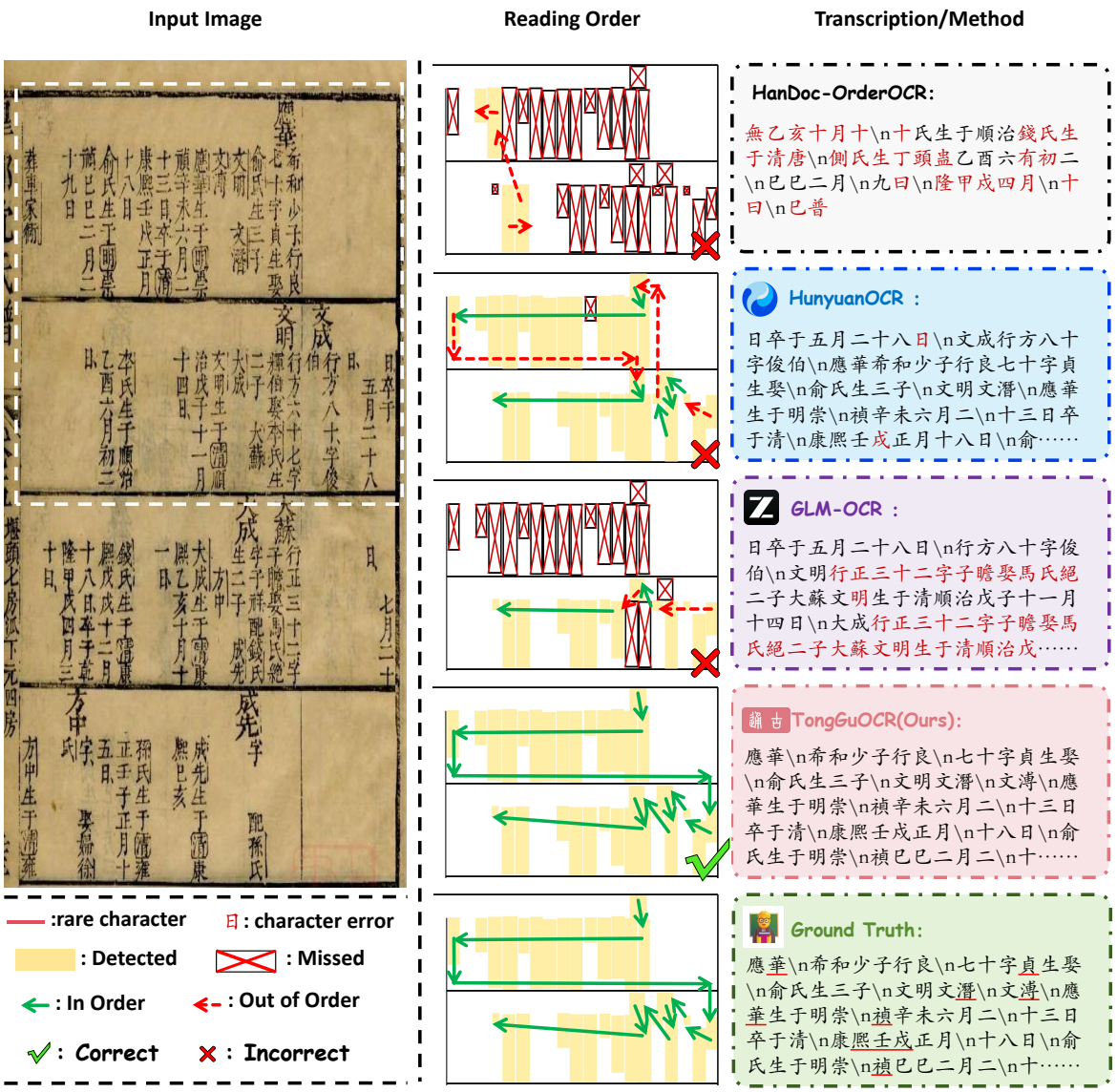

Fig. 1 Recognition results illustrating the challenges of Chinese historical document OCR. The white dashed box in the input image marks the region used for the reading-order and transcription comparison. This example contains a table-like layout, a nontrivial reading order, and rare historical characters. We compare TongGuOCR with representative OCR systems, including HanDoc-OrderOCR [14], HunyuanOCR [15], and GLM-OCR [16]. Arrows visualize the reading paths aligned to the source region, while red characters indicate recognition errors or order-related mismatches. TongGuOCR follows the natural reading sequence of the page and accurately transcribes the selected content in this case.

recognition blocks from text-line boxes and refines them into OCR-friendly crops. These recognition blocks preserve character visibility while retaining sufficient local context.

- We introduce a **Token-Augmented Recognition** module with two complementary designs. Character-level vocabulary expansion improves the representation of rare characters, and line-to-line transition modeling provides spatial guidance during transcription.
- Experiments on two Chinese historical document OCR benchmarks show that TongGuOCR outperforms representative traditional task-specific OCR models, general-purpose MLLMs,

and OCR-oriented MLLMs, achieving AR scores of 97.93 and 93.76 and RO-ED scores of 1.51 and 3.49 on MTHv2 and M^{5}HisDoc, respectively. Ablation studies verify the effectiveness of both modules and the three-stage SFT strategy.

2 Related Work

2.1 Traditional Historical Document OCR

OCR for Chinese historical documents has traditionally been studied under a modular document-analysis paradigm. Instead of directly transcribing a full page in an end-to-end manner, earlier approaches usually decompose the task into stages such as image preprocessing, layout analysis, text-line or character localization, recognition, reading-order recovery, and post-correction [13, 17–19, 24]. Within this paradigm, prior studies have developed OCR systems for traditional Chinese texts [13, 17, 19], investigated dense character detection and localization in historical document pages [24, 25], and jointly considered layout analysis, character detection, and recognition for historical document digitization [18]. Recent datasets and benchmarks, such as M^{5}HisDoc and HisDoc1B, further support research on multi-style and large-scale Chinese historical document analysis [26, 27]. These works provide explicit structural modeling of document pages and enable fine-grained processing at the region, line, or character level.

Despite their explicit structural modeling, traditional historical-document OCR systems commonly treat individual text lines or characters as separate recognition inputs. This fine-grained formulation preserves local character resolution but provides limited context across neighboring lines. Moreover, traditional recognizers usually rely on predefined character sets, while page-level reading sequences are often reconstructed through separately designed geometric rules. These characteristics make rare characters and complex local line transitions difficult to handle. They motivate OCR frameworks that retain explicit layout processing while using a more suitable recognition granularity, more flexible character representations, and explicit spatial guidance during transcription.

2.2 Large Models for Document OCR

Beyond traditional approaches, recent MLLMs have substantially advanced document OCR and document parsing. General-purpose MLLMs perform OCR through visual prompting and instruction following, benefiting from broad visual-language pretraining and semantic reasoning ability [20, 21, 28–31]. OCR-oriented MLLMs are designed specifically for document recognition and parsing. They include models that directly generate plain text, Markdown, or structured representations from document images [15, 22, 32–35], as well as MLLM-based systems that integrate layout analysis, recognition, reading-order prediction, and structure reconstruction [16, 23, 36–40]. These methods have shown strong performance on modern document images and complex document parsing tasks.

Nevertheless, these systems are primarily optimized for general document domains and do not explicitly account for characteristics of Chinese historical materials, such as complex layouts, historical characters, and nontrivial line transitions. This domain mismatch limits their reliability in faithful historical-document transcription and motivates task-specific adaptation.

3 Methods

As illustrated in Figure 2, TongGuOCR consists of two modules. Layout-Aware Preprocessing (Section 3.1) converts a full page into an ordered sequence of OCR-friendly recognition blocks that preserve local context while reducing interference from unrelated regions. Token-Augmented Recognition (Section 3.2) then transcribes each block, improving rare-character representation and providing spatial guidance during decoding. TongGuOCR employs a three-stage supervised fine-tuning strategy to adapt the recognition model to historical documents and align its training inputs with the recognition-block crops used during inference (Section 3.3).

3.1 Layout-Aware Preprocessing

A key design choice in TongGuOCR is the recognition granularity. Whole-page input requires aggressive resizing that blurs small characters [41],

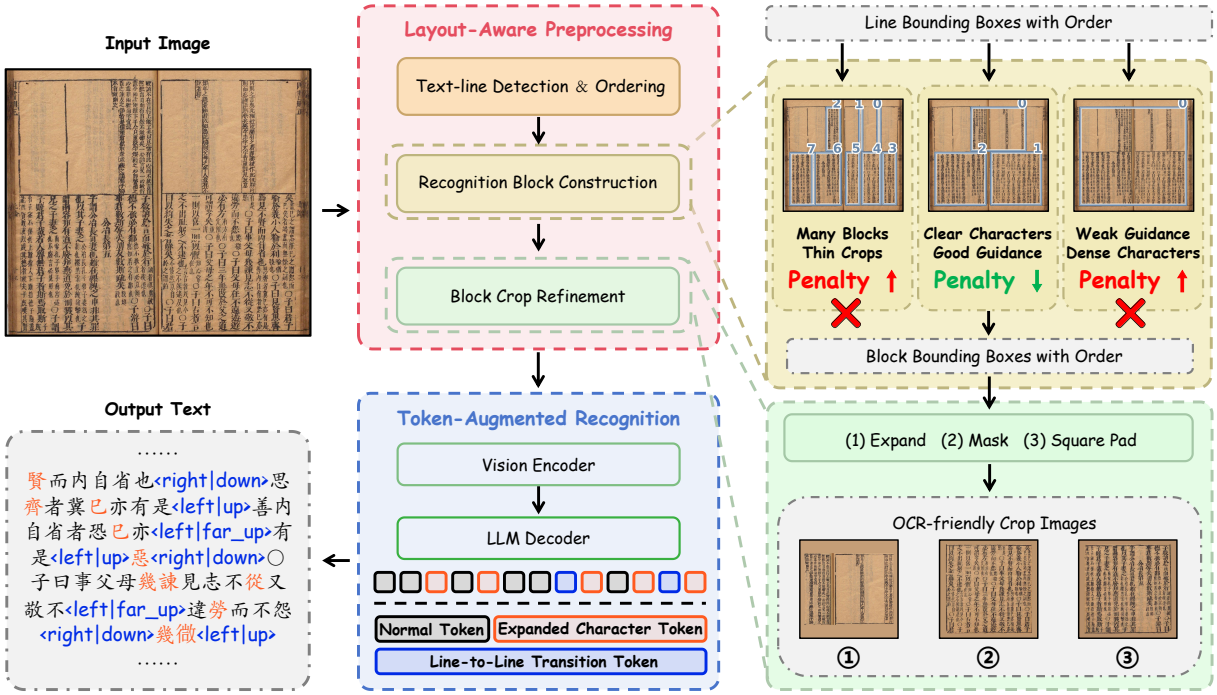

Fig. 2 Overview of TongGuOCR. The framework consists of two modules. Layout-Aware Preprocessing obtains text-line boxes, constructs recognition blocks, and refines them into OCR-friendly crops. Token-Augmented Recognition then processes these crops using character-level vocabulary expansion and line-to-line transition modeling, after which the resulting transcriptions are merged into the page-level output. The enlarged panels on the right illustrate how block construction and crop refinement affect OCR inputs. Over-segmented candidates produce many narrow crops with limited context, while under-segmented candidates contain dense characters and provide weak layout guidance; both receive higher penalties. Balanced blocks preserve clearer characters and useful local layout context, resulting in a lower penalty and more OCR-friendly inputs.

whereas line-level recognition yields elongated crops, loses neighboring context, and needs one forward pass per line [42]. Recognition blocks offer an intermediate granularity, but grouping lines is nontrivial, since visual adjacency need not match reading-order adjacency (e.g., interlinear annotations) and the ideal block size is layout-dependent. We therefore restrict each block to consecutive lines in the reading order and organize the grouping into three steps, detailed below: Candidate Block Construction, Optimal Segmentation Selection, and Block Crop Refinement.

3.1.1 Candidate Block Construction

Text-line boxes and reading order. We first obtain the text-line boxes of a page together with their reading order, denoted $L = [l_1, l_2, \ldots, l_n]$. During training, the boxes and their order are taken from annotations; during inference, they are produced by an external historical-document layout module [17]. This order determines which lines may be grouped consecutively and is preserved throughout the resulting block sequence.

Candidate blocks and segmentations. For a start index i and an exclusive end boundary j, we define a candidate recognition block as

$$B_{i,j} = (l_i, l_{i+1}, \ldots, l_{j-1}), \qquad 1 \le i < j \le n+1. \tag{1}$$

For example, $B_{3,6} = (l_3, l_4, l_5)$. Restricting each candidate to consecutive lines preserves the reading order in L and avoids grouping spatially adjacent lines that are discontinuous in the sequence. A valid segmentation $S = [B_1, \ldots, B_K]$ selects non-overlapping candidate blocks that jointly cover L, so that every text line is assigned to exactly one block without changing its order. We denote the set of all such segmentations by $\mathcal{S}(L)$.

3.1.2 Optimal Segmentation Selection

Penalty-based segmentation objective. Let $\mathcal{P}(S)$ denote the total layout penalty of a complete segmentation S. We select the valid segmentation with the lowest total penalty:

$$S^* = \arg\min_{S \in \mathcal{S}(L)} \mathcal{P}(S), \tag{2}$$

where

$$\begin{aligned}\mathcal{P}(S) = \lambda_{\text{blk}} &\sum_{t=1}^{K} p_{\text{blk}}(B_t) \\ &+ \lambda_{\text{adj}} \sum_{t=1}^{K-1} p_{\text{adj}}(B_t, B_{t+1}) \\ &+ \lambda_{\text{page}} p_{\text{page}}(K).\end{aligned} \tag{3}$$

The three penalty groups evaluate complementary aspects of a segmentation. The block-level penalty p_{blk} measures whether each candidate balances character visibility and local layout context, corresponding to the different granularities illustrated in Figure 2. The adjacent-block penalty p_{adj} discourages geometrically unstable boundaries, such as severe overlap or containment between successive blocks. The page-level penalty p_{page} regularizes the total number of blocks to avoid both whole-page grouping and excessive fragmentation. The non-negative coefficients λ_{blk}, λ_{adj}, and λ_{page} control their relative contributions.

Dynamic programming solution. The consecutive-interval representation gives the segmentation problem an ordered optimal substructure: a partial segmentation ending at boundary j can be formed by appending a candidate $B_{i,j}$ to a predecessor that ends at i. We therefore use dynamic programming over interval boundaries and block counts to obtain the minimum-penalty segmentation within the retained candidate space. After backtracking, a lightweight local merge step absorbs isolated tiny blocks into geometrically compatible neighboring blocks to reduce unstable fragmentation. Detailed geometric cues, candidate pruning, the dynamic-programming recurrence, backtracking, and local merging are provided in supplementary material.

3.1.3 Block Crop Refinement

Each selected block is finally converted into an OCR-friendly crop with reduced cross-block interference and more stable geometry, as shown in the lower-right panel of Figure 2. Existing OCR systems commonly resize document regions using high-resolution or dynamic-resolution visual encoders [43–45] and regularize them into square or fixed-size inputs [46]. Consequently, tightly bounded or highly elongated crops may lose boundary context or undergo excessive downsampling. We therefore expand each selected block box by a small margin, mask overlapping text lines belonging to other blocks, and place the processed crop on a padded square canvas while preserving its aspect ratio.

Boundary expansion. We first enlarge each selected block box by a small margin, so that characters near the block boundary retain sufficient surrounding context after cropping.

Non-target line masking. For block B_t, let M_t^{own} denote the mask of its own text lines and M_t^{nt} the mask of overlapping non-target lines. The region to be removed is

$$M_t = M_t^{\text{nt}} \setminus M_t^{\text{own}}. \tag{4}$$

Masked pixels are filled with a locally estimated background color.

Aspect-ratio-preserving padding. If the resulting crop has width w_t and height h_t, it is centered on a square canvas with side length

$$s_t = \alpha \max(w_t, h_t), \tag{5}$$

where α is a fixed padding scale. The refined crops preserve their original aspect ratios and block order before being passed to the recognition module. Further implementation details are provided in the supplementary material.

3.2 Token-Augmented Recognition

The recognition module produces a token sequence rather than isolated character predictions. Therefore, the construction of the transcription target affects both character generation and the spatial organization of the output sequence. We augment the target sequence at two complementary levels. First, we expand the tokenizer with single Chinese character tokens so that rare

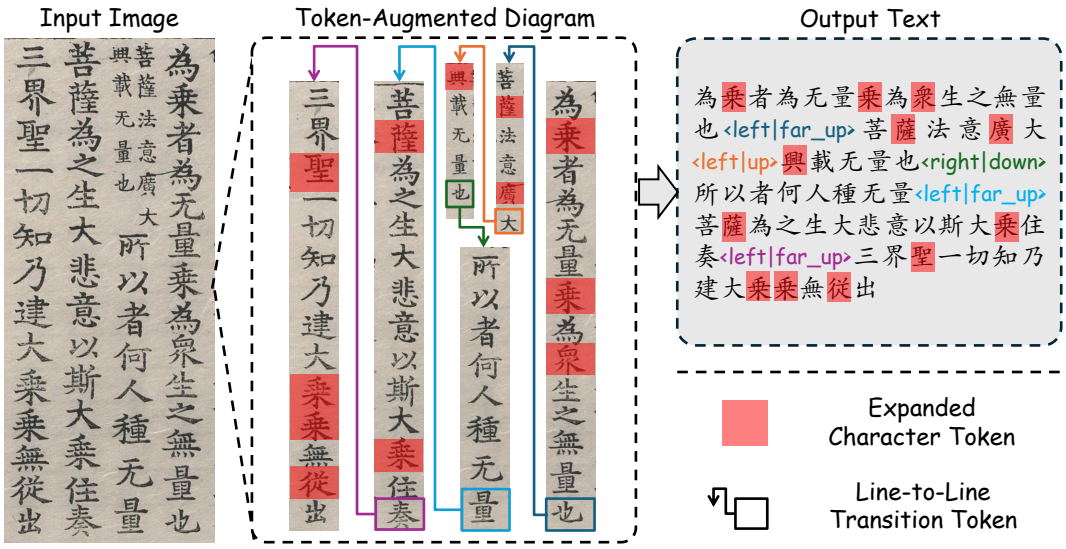

Fig. 3 Illustration of the Token-Augmented Recognition target construction. The transcription is represented as an augmented sequence containing ordinary text tokens, expanded character tokens, and line-to-line transition tokens. The highlighted Chinese characters indicate previously fragmented characters that are represented as single tokens after vocabulary expansion. Transition tokens, such as `<left|far_up>` and `<right|down>`, encode the coarse spatial displacement from the end of one text line to the beginning of the next.

characters can be represented more compactly. Second, we insert line-to-line transition tokens between adjacent line transcriptions. These tokens encode the coarse spatial displacement from the end of one line to the beginning of the next, providing auxiliary spatial cues during autoregressive decoding. Figure 3 illustrates the resulting augmented sequence construction.

3.2.1 Character-Level Vocabulary Expansion

Faithful transcription of Chinese historical documents requires character-level output. However, tokenizers inherited from pretrained MLLMs are usually optimized for modern text and often lack standalone tokens for rare Chinese characters. A single rare character may therefore be decomposed into multiple byte-level or subword tokens, requiring several autoregressive decoding steps to transcribe one visual character instance. Representing such a character as a single token makes its visual-to-output correspondence more direct and shortens its decoding path, potentially reducing character-level generation errors and reliance on contextual language priors.

Candidate character mining. Let $\mathcal{V}_0$ denote the vocabulary of the original tokenizer, and let $\mathrm{Tok}_0(\cdot)$ denote its tokenization function. We train an auxiliary SentencePiece tokenizer with the byte-pair encoding (BPE) model [47] on the training corpus. The validation and test splits are excluded. This auxiliary tokenizer is used only to mine candidate characters, rather than to replace the original tokenizer. We remove special tokens such as `<unk>`, `<s>`, and `</s>`, strip the SentencePiece whitespace marker when present, and retain candidate characters from the Unicode CJK Unified Ideographs, CJK Compatibility Ideographs, and related extension blocks.[1] Let $\mathcal{C}_{\mathrm{CJK}}$ denote this candidate character set.

Character selection criterion. From these candidates, we add only characters satisfying the following condition:

$$\mathcal{A} = \{c \mid |c| = 1,\ c \in \mathcal{C}_{\mathrm{CJK}},\ c \notin \mathcal{V}_0,\ |\mathrm{Tok}_0(c)| > 1\}. \tag{6}$$

That is, a new character token is added only when it is a single character, is absent from the original vocabulary, and would otherwise be fragmented into multiple tokens by the original tokenizer. After vocabulary expansion, the model embedding

[1] CJK stands for Chinese, Japanese, and Korean. For Unicode's treatment of Han ideographs across these writing systems, see https://www.unicode.org/faq/han_cjk.html.

matrix is resized using the default model resizing routine, and the newly added embeddings are optimized during training.

3.2.2 Line-to-Line Transition Modeling

Layout analysis and region cropping reduce page-level complexity but do not eliminate ambiguity in transitions between text lines, since both full pages and cropped regions may contain multiple columns, interlinear annotations, or other irregular arrangements. After transcribing one line, the model must locate the visual region of the next line, and errors here cause skipped or repeated lines and cross-line confusion. To supervise this process, we insert a discrete transition token between adjacent line transcriptions that encodes the coarse spatial displacement from the end of one line to the beginning of the next, guiding the decoder toward the next relevant region without requiring precise coordinates.

Transition token insertion. Given a dataset-annotated sequence of text lines $L = [l_1, l_2, \ldots, l_n]$, we augment the transcription target by inserting a line-to-line transition token between each pair of adjacent line transcriptions:

$$Y_{\text{trans}} = T(l_1) \oplus r_1 \oplus T(l_2) \oplus \cdots \oplus r_{n-1} \oplus T(l_n), \quad (7)$$

where $T(\cdot)$ denotes tokenization with the expanded tokenizer, $\oplus$ denotes sequence concatenation, and r_i encodes the coarse spatial displacement from the end of l_i to the beginning of l_{i+1}.

Line-to-line displacement computation. For each adjacent pair of text lines, r_i is determined by the displacement from the end anchor of the current line to the start anchor of the next line:

$$\Delta_i = \left(x_{i+1}^{\text{start}} - x_i^{\text{end}}, y_{i+1}^{\text{start}} - y_i^{\text{end}}\right). \quad (8)$$

The start and end anchors are defined as the centers of the first and last dataset-annotated characters in the corresponding line, respectively. This definition aligns the transition token with the actual spatial movement between successive line transcriptions. The horizontal and vertical offsets are normalized by the image width and height. Under the image coordinate system, the sign of the horizontal offset determines `left` or `right`, and the sign of the vertical offset determines `up` or `down`.

Coarse displacement quantization. To convert continuous line-to-line displacements into discrete transition tokens, we apply a fixed coarse quantization rule. The goal of this quantization is not to model precise geometry, but to provide a compact set of stable spatial categories for auxiliary supervision at line boundaries. Let δ denote a near-zero threshold and τ denote a long-range threshold. For each axis, offsets whose magnitudes are smaller than δ are omitted, whereas offsets whose magnitudes exceed τ are assigned the prefix `far_`. If both horizontal and vertical offsets are negligible, we retain the axis with the larger normalized displacement to ensure that each adjacent line pair receives a valid transition token. The horizontal and vertical label sets are

$$\mathcal{R}_x = \{\texttt{left}, \texttt{right}, \texttt{far_left}, \texttt{far_right}\}, \quad (9)$$

and

$$\mathcal{R}_y = \{\texttt{up}, \texttt{down}, \texttt{far_up}, \texttt{far_down}\}. \quad (10)$$

Transition vocabulary construction. The final transition token vocabulary is constructed as

$$\begin{aligned} \mathcal{R} = &\{\langle x \rangle \mid x \in \mathcal{R}_x\} \\ &\cup \{\langle y \rangle \mid y \in \mathcal{R}_y\} \\ &\cup \{\langle x|y \rangle \mid x \in \mathcal{R}_x,\ y \in \mathcal{R}_y\}. \end{aligned} \quad (11)$$

This construction yields a compact set of 24 possible transition tokens. For instance, the transition from the bottom of one vertical column to the top of the neighboring column can be represented by `<left|far_up>`, while a short movement into an annotation region or a return to the main text stream can be represented by tokens such as `<right|down>` or `<left|up>`. Thus, each token describes the coarse relative location of the next line.

By predicting transition tokens jointly with text, the decoder learns to associate each line boundary with the approximate location of the next text line. These tokens therefore provide auxiliary spatial supervision during transcription, supporting more stable and accurate output generation.

Table 1 Dataset annotation source and split usage. SFT, Select, and Eval denote supervised fine-tuning, checkpoint selection, and final evaluation, respectively.

Dataset	Annotation source	Train	Val	Test
HisDoc1B	Semi-automatic	3.16M / SFT	–	–
MTHv2	Human	2,399 / SFT	–	800 / Eval
M^{5}HisDoc	Human	2,000 / SFT	1,000 / Select	1,000 / Eval

3.3 Training Strategy

TongGuOCR is optimized through three successive SFT stages. Stage 1 performs coarse domain adaptation with diverse page-level samples and develops broad familiarity with Chinese historical documents. Stage 2 refines page-level recognition with human-annotated data. Stage 3 uses recognition-block crops generated by Layout-Aware Preprocessing, thereby aligning the training and inference inputs. All three stages use the same augmented target representation and autoregressive objective.

Given an input image I and an augmented transcription sequence $Y = (y_1, y_2, \ldots, y_T)$, the model is optimized with the standard autoregressive language modeling objective:

$$\mathcal{L}_{\mathrm{LM}} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(y_t \mid I, y_{<t}). \qquad (12)$$

The target sequence Y contains ordinary text tokens and line-to-line transition tokens; text generation and transition-token prediction are therefore optimized jointly under the same next-token prediction objective.

4 Experiments

4.1 Datasets

We conduct experiments on three publicly released Chinese historical document datasets: HisDoc1B [27], MTHv2 [18], and M^{5}HisDoc [26]. HisDoc1B serves as a large-scale dataset for SFT, while M^{5}HisDoc and MTHv2 are used for SFT and evaluation. Table 1 summarizes the annotation source and split usage of each dataset.

HisDoc1B. This large-scale Chinese historical document dataset contains 40,281 books, 3,163,330 images, 1,082,544,808 characters, and 30,615 character categories. It was constructed through a semi-automatic annotation pipeline designed for large-scale historical document collection and annotation. In our experiments, HisDoc1B serves as a large-scale training set that exposes the model to diverse historical document styles, page layouts, character forms, and transcription lengths.

MTHv2. Built from the Tripitaka Koreana in Han and Multiple Tripitaka in Han collections, this dataset contains 3,199 document images, 1,081,678 characters, and 6,733 character categories. In our experiments, we follow the official train/test split, which consists of 2,399 training images and 800 testing images. MTHv2 does not provide a validation split.

M^{5}HisDoc. This multi-style Chinese historical document dataset contains 4,000 images, 403,824 text lines, 4,367,392 characters, and 16,151 character categories. It covers diverse layouts, document types, calligraphic styles, and backgrounds. In our experiments, we follow the regular setting and use the official split, with 2,000 images for training, 1,000 images for validation, and 1,000 images for evaluation.

Note that HisDoc1B does not include any samples from the test sets of MTHv2 or M^{5}HisDoc.

4.2 Evaluation Metrics

The OCR-related metrics are implemented using a modified version of the evaluation protocol from MCS-Bench [48]. Unlike MCS-Bench, we do not convert model outputs to Simplified Chinese before evaluation. For each metric, scores are computed on plain textual transcriptions for individual samples, averaged across all evaluation samples, and reported as percentages.

Accuracy and correct rate (AR and CR). We use AR and CR as the primary OCR accuracy metrics. For a given sample, let G denote the reference transcription and P denote the predicted transcription. Based on the Levenshtein alignment between G and P, let S, D, and I be the numbers

of substitutions, deletions, and insertions, respectively. Let $N = |G|$ be the number of characters in the reference transcription. AR and CR are defined as

$$\mathrm{AR} = \frac{N - S - D - I}{N}, \tag{13}$$

and

$$\mathrm{CR} = \frac{N - S - D}{N}. \tag{14}$$

AR penalizes substitutions (S), deletions (D), and insertions (I), whereas CR penalizes only S and D and is therefore unaffected by insertion errors. Notably, AR can take negative values: AR < 0 when $S+D+I > N$, meaning that the total number of substitution, deletion, and insertion errors exceeds the number of ground-truth characters.

Normalized edit distance (NED). We report NED:

$$\mathrm{NED} = \frac{\mathrm{EditDistance}(P, G)}{\max(|P|, |G|)}, \tag{15}$$

where $\mathrm{EditDistance}(\cdot, \cdot)$ denotes the Levenshtein edit distance. Lower NED indicates better sequence-level agreement.

Character-inventory precision, recall, and F1. We report character-inventory precision, recall, and F1 score as auxiliary coverage metrics. Let C_P and C_G denote the sets of unique characters appearing in the prediction and reference, respectively. They are defined as

$$\mathrm{Precision} = \frac{|C_P \cap C_G|}{|C_P|}, \tag{16}$$

$$\mathrm{Recall} = \frac{|C_P \cap C_G|}{|C_G|}, \tag{17}$$

and

$$\mathrm{F1} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{18}$$

These set-based metrics measure character-inventory coverage, but they do not account for character order or repeated occurrences. Thus, F1 reflects the overlap between the predicted and reference character inventories, whereas AR evaluates occurrence-level accuracy over the full transcription sequence by penalizing substitutions, deletions, and insertions.

Character-level BLEU. Character-level BLEU serves as an auxiliary sequence-similarity metric. Both the reference and prediction are segmented into character sequences, and sentence-level BLEU is computed with smoothing.

Reading-order edit distance (RO-ED). To explicitly evaluate reading-order preservation, we follow the reading-order evaluation protocol of OmniDocBench [49], which measures reading order using normalized edit distance over matched textual components, and report a line-level RO-ED. RO-ED is computed on sequences of matched text-line indices. For each page, the ground-truth transcription is decomposed into an ordered list of text lines, and each line is assigned a reading-order index, forming the reference sequence $\mathbf{o}^{gt} = [1, 2, \ldots, n]$. The predicted text lines are first matched to ground-truth lines according to textual similarity. The ground-truth indices of the matched lines are then sorted according to their positions in the model output, yielding the predicted order sequence $\mathbf{o}^{pred} = [\pi_1, \pi_2, \ldots, \pi_m]$, where π_j denotes the ground-truth reading-order index of the j-th matched predicted line.

RO-ED is computed as the normalized edit distance between the two index sequences:

$$\text{RO-ED} = \frac{\mathrm{ED}_{seq}(\mathbf{o}^{gt}, \mathbf{o}^{pred})}{\max(|\mathbf{o}^{gt}|, |\mathbf{o}^{pred}|)}, \tag{19}$$

where ED_{seq} denotes Levenshtein edit distance over discrete line indices. Lower RO-ED indicates better reading-order preservation, with 0 corresponding to a perfectly ordered prediction. We report the average RO-ED over all test pages.

Higher values are better for AR, CR, precision, recall, F1, and BLEU, whereas lower values are better for NED and RO-ED.

4.3 Implementation Details

Runtime environment. Experiments were conducted under Python 3.11.13 and PyTorch 2.7.1 with Ascend NPU support, implemented with the LLaMA-Factory [50], and executed on a server equipped with eight Huawei Ascend 910B2 NPUs, each with 64 GB HBM.

Model configuration. We adopt GLM-OCR [16] as our foundation model. We first extend the original tokenizer with both the newly added single-character Chinese tokens and the line-to-line transition tokens, and then expand the model token embeddings accordingly. The original token embeddings are inherited from the pretrained

Table 2 Implementation settings of the three-stage training strategy. MTH denotes MTHv2, M5H denotes M^{5}HisDoc, and H1B denotes HisDoc1B.

Setting	Stage 1	Stage 2	Stage 3
Training data	H1B, MTH, M5H	MTH, M5H	MTH, M5H
Input granularity	Page	Page	Block
Max image resolution	768×768	1024×1024	1024×1024
Max sequence tokens	6144	8192	8192
Learning rate	3×10^{-5}	1×10^{-5}	1×10^{-5}
Epochs	3	10	15

model, while the newly added embedding rows are initialized with the default strategy. For line-to-line transition quantization, we use fixed heuristic thresholds of $\delta = 0.01$ and $\tau = 0.5$ after normalization. Here, δ suppresses near-zero displacements caused by minor localization variations, while τ marks an axis-wise movement spanning more than half of the page as a long-range transition. These thresholds are used throughout all experiments.

For Layout-Aware Preprocessing, we set $\lambda_{\text{blk}} = 2$, $\lambda_{\text{adj}} = 2$, and $\lambda_{\text{page}} = 1$ throughout all experiments.

Training configuration. We adopt full-parameter SFT in all stages; that is, the vision encoder, multimodal projector, language model, and newly added token embeddings are all updated. All training stages share the same optimization configuration. We use the AdamW optimizer [51], an adaptive optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. Training follows a cosine decay schedule with a warmup ratio of 0.1, and gradients are clipped with a maximum norm of 1.0. Scaled dot-product attention is used as the attention backend. All training stages were conducted using bfloat16 (bf16) precision.

The stage-specific training configurations are summarized in Table 2.

Stage 1: Coarse domain adaptation. Stage 1 performs coarse domain adaptation using page-level training samples from HisDoc1B, MTHv2, and M^{5}HisDoc. The full HisDoc1B training set is used without subsampling. This stage adopts a maximum image resolution of 768×768, a maximum sequence length of 6144 tokens, a learning rate of 3×10^{-5}, and 3 training epochs.

Stage 2: High-quality page-level refinement. Stage 2 performs high-quality page-level refinement using the human-annotated training splits of MTHv2 and M^{5}HisDoc. It adopts a maximum image resolution of 1024×1024, a maximum sequence length of 8192 tokens, a learning rate of 1×10^{-5}, and 10 training epochs.

Stage 3: Block-level inference alignment. Stage 3 performs block-level inference alignment using recognition-block samples generated from the MTHv2 and M^{5}HisDoc training splits. These samples are produced by the same Layout-Aware Preprocessing pipeline employed during inference. This stage uses the same maximum image resolution, sequence length, and learning rate as Stage 2, and is trained for 15 epochs.

Checkpoint selection. Checkpoint selection is based on performance on the official validation split of M^{5}HisDoc. The final Stage 1 checkpoint at epoch 3 is used to initialize Stage 2, the validation-selected Stage 2 checkpoint at epoch 5 is used to initialize Stage 3, and the validation-selected Stage 3 checkpoint at epoch 3 is used for the main experimental results.

Inference procedure. During inference, the full page is first processed by Layout-Aware Preprocessing, which produces an ordered sequence of OCR-friendly crops. Each crop is then recognized autoregressively. Line-to-line transition tokens in each decoded sequence are replaced with newline characters to preserve line boundaries, and the resulting text transcriptions are concatenated in the sequence provided by Layout-Aware Preprocessing to obtain the final page-level transcription.

4.4 Main Results

Compared methods. The comparison includes representative methods from three categories: traditional task-specific OCR models, general-purpose MLLMs, and OCR-oriented MLLMs. Some traditional task-specific baselines use text-line crops constructed from dataset-provided annotations, whereas the MLLM-based methods operate on full-page images or use OCR-oriented processing pipelines.

Table 3 Performance comparison on MTHv2. AR, CR, NED, F1, P, R, BLEU, and RO-ED denote accuracy rate, correct rate, normalized edit distance, F1 score, precision, recall, BLEU, and reading-order edit distance, respectively. A dash denotes a result that was not reported. Higher values are better for AR, CR, F1, P, R, and BLEU, while lower values are better for NED and RO-ED. Bold and underlined values denote the best and second-best results among all compared methods, respectively. * indicates results reported under the text-line recognition setting and taken from ZCTRN [52]; in this setting, input line images are constructed from the dataset-provided text-line annotations. † indicates results reported in the MTHv2 paper [18].

Method	Venue / Source	MTHv2							
		AR↑	CR↑	NED↓	F1↑	P↑	R↑	BLEU↑	RO-ED↓
Traditional task-specific OCR models									
RAN [53]*	PR 2020	91.56	91.79	-	-	-	-	-	-
CRNN [54]*	T-PAMI 2016	96.94	97.15	-	-	-	-	-	-
ZCTRN [52]*	ICDAR 2021	<u>97.42</u>	97.62	-	-	-	-	-	-
Ma et al. [18]†	ICFHR 2020	95.52	96.07	-	-	-	-	-	-
HanDoc-OrderOCR [14]	AAAI 2024	95.27	95.84	4.72	96.72	97.24	96.28	92.22	4.71
Liu et al. [17]	NSR 2023	96.30	97.18	3.61	96.94	96.42	97.52	93.99	3.19
General-purpose MLLMs									
Qwen3-VL-2B [20]	arXiv 2025	-376.72	71.61	54.71	76.94	88.51	72.37	40.12	32.43
Qwen3-VL-8B [20]	arXiv 2025	-29.34	82.70	33.38	85.43	90.51	83.27	61.33	21.06
Qwen3.5-0.8B [28]	arXiv 2026	-69.63	70.59	38.38	79.24	86.09	75.39	53.38	31.99
Qwen3.5-9B [28]	arXiv 2026	62.17	80.74	23.05	86.63	89.51	84.68	69.68	18.97
InternVL3.5-2B [21]	arXiv 2025	-126.62	59.13	51.70	74.48	91.90	66.25	44.66	44.89
InternVL3.5-8B [21]	arXiv 2025	-38.99	72.65	33.90	81.22	92.13	74.53	57.53	29.88
Qwen3.7-Plus [29]	Official API	81.24	88.06	15.73	88.76	88.30	89.59	77.48	15.95
MiniMax M3 [55]	Official API	67.54	75.15	27.69	81.65	85.47	79.40	63.48	26.01
Kimi K2.6 [56]	Official API	86.40	90.66	11.79	92.14	92.40	92.10	81.97	10.57
GPT-5.5 [31]	Official API	70.18	74.38	28.59	80.61	85.30	78.00	64.26	28.03
Gemini 3.1 Pro [30]	Official API	78.09	84.44	19.08	87.79	88.66	87.83	75.33	19.77
OCR-oriented MLLMs									
DeepSeek-OCR 2 [22]	arXiv 2026	-540.58	26.55	83.83	41.69	71.87	33.66	13.56	68.81
DeepSeek-OCR [32]	arXiv 2025	-494.24	63.41	60.37	73.25	86.37	66.46	35.81	44.13
FireRed-OCR [34]	arXiv 2026	-198.23	82.38	26.66	85.78	91.40	83.04	66.73	17.24
dots.ocr [33]	arXiv 2025	-3.27	84.11	23.65	87.71	90.84	86.02	69.09	16.43
Qianfan-OCR [35]	arXiv 2026	62.79	86.90	18.23	88.35	88.86	88.48	73.67	16.42
HunyuanOCR [15]	arXiv 2025	77.80	<u>97.95</u>	<u>3.13</u>	<u>97.63</u>	<u>98.06</u>	<u>97.53</u>	<u>95.07</u>	<u>1.97</u>
MonkeyOCR-1.2B [37]	arXiv 2025	-432.67	35.71	77.76	46.88	55.78	43.79	14.16	52.89
MonkeyOCR-3B [37]	arXiv 2025	-184.49	53.07	58.15	62.00	66.40	59.59	28.88	46.35
MinerU2.5 [38]	arXiv 2025	48.68	58.65	47.35	66.93	62.78	75.14	44.60	39.44
PaddleOCR-VL-1.5 [36]	arXiv 2026	47.36	79.90	28.10	87.15	89.76	85.76	72.07	22.75
MinerU2.5-Pro [23]	arXiv 2026	75.95	83.85	20.48	87.02	86.05	88.50	71.25	20.17
GLM-OCR [16]	arXiv 2026	69.47	74.56	28.88	83.50	82.84	84.96	62.72	22.34
PaddleOCR-VL-1.6 [39]	arXiv 2026	72.25	89.08	15.59	92.27	93.36	92.19	85.04	14.29
TongGuOCR	This work	**97.93**	**98.33**	**2.05**	**98.03**	**98.07**	**98.02**	**96.23**	**1.51**

Note. Negative AR values occur when insertion errors are sufficiently severe that $S + D + I > N$, indicating over-generation rather than a computational error; see Section 4.2.

Results on MTHv2. As shown in Table 3, on MTHv2, strong historical-document baselines already achieve high recognition accuracy, indicating that the benchmark is relatively saturated. TongGuOCR nevertheless reaches 97.93 AR, 98.33 CR, and 2.05 NED. Relative to the best competing score for each primary metric, it improves AR from 97.42 to 97.93 and CR from 97.95 to 98.33, while reducing NED from 3.13 to 2.05. It also achieves the best character-inventory metrics and BLEU score and reduces RO-ED to 1.51. These results suggest that TongGuOCR achieves the most accurate character recognition, the highest transcription fidelity, and the best preservation of reading order among all compared methods on MTHv2.

Results on M^{5}HisDoc. As shown in Table 4, on M^{5}HisDoc, the advantage is more pronounced. This benchmark contains more diverse layouts, visual styles, backgrounds, and historical character forms. TongGuOCR achieves 93.76 AR, 94.46 CR, and 6.15 NED, compared with the best competing scores of 92.10, 92.29, and 10.43, respectively. It also obtains the best character-inventory

Table 4 Performance comparison on M^{5}HisDoc. AR, CR, NED, F1, P, R, BLEU, and RO-ED denote accuracy rate, correct rate, normalized edit distance, F1 score, precision, recall, BLEU, and reading-order edit distance, respectively. A dash denotes a result that was not reported. Higher values are better for AR, CR, F1, P, R, and BLEU, while lower values are better for NED and RO-ED. Bold and underlined values denote the best and second-best results among all compared methods, respectively. * indicates results reported under the text-line recognition setting and taken from the M^{5}HisDoc report [26]; in this setting, input line images are constructed from the dataset-provided text-line annotations.

Method	Venue / Source	M^{5}HisDoc							
		AR↑	CR↑	NED↓	F1↑	P↑	R↑	BLEU↑	RO-ED↓
Traditional task-specific OCR models									
CRNN [54]*	T-PAMI 2016	91.36	91.49	-	-	-	-	-	-
Ma et al. [18]*	ICFHR 2020	<u>92.10</u>	<u>92.29</u>	-	-	-	-	-	-
ZCTRN [52]*	ICDAR 2021	88.50	88.79	-	-	-	-	-	-
ASTER [57]*	T-PAMI 2018	87.70	87.98	-	-	-	-	-	-
NRTR [58]*	ICDAR 2019	83.02	85.43	-	-	-	-	-	-
RobustScanner [59]*	ECCV 2020	90.78	90.99	-	-	-	-	-	-
Peng et al. [60]*	TMM 2022	88.35	88.38	-	-	-	-	-	-
HanDoc-OrderOCR [14]	AAAI 2024	54.45	55.71	45.45	65.68	73.17	63.18	40.42	32.59
Liu et al. [17]	NSR 2023	89.47	91.07	<u>10.43</u>	<u>92.02</u>	91.23	<u>92.92</u>	<u>83.74</u>	7.71
General-purpose MLLMs									
Qwen3-VL-2B [20]	arXiv 2025	-100.41	49.37	63.55	59.56	77.77	54.51	29.42	42.70
Qwen3-VL-8B [20]	arXiv 2025	-5.07	64.64	45.11	72.48	81.19	69.48	45.86	32.64
Qwen3.5-0.8B [28]	arXiv 2026	-63.41	52.61	53.58	64.28	74.19	60.66	35.80	39.47
Qwen3.5-9B [28]	arXiv 2026	53.01	67.42	35.80	75.73	79.86	73.54	51.86	30.02
InternVL3.5-2B [21]	arXiv 2025	-130.28	37.82	68.75	58.50	82.82	50.94	28.32	58.80
InternVL3.5-8B [21]	arXiv 2025	-27.80	57.27	46.65	68.08	82.44	62.30	41.28	35.41
Qwen3.7-Plus [29]	Official API	64.55	70.77	32.86	76.73	80.02	75.27	55.92	24.86
MiniMax M3 [55]	Official API	5.97	39.02	64.07	56.66	68.48	51.54	27.84	51.47
Kimi K2.6 [56]	Official API	45.27	73.84	31.71	80.98	84.08	79.62	60.60	22.55
GPT-5.5 [31]	Official API	45.01	51.34	51.88	63.88	70.70	61.05	39.33	46.45
Gemini 3.1 Pro [30]	Official API	61.05	67.76	35.99	76.41	79.68	74.58	53.80	29.33
OCR-oriented MLLMs									
DeepSeek-OCR 2 [22]	arXiv 2026	-357.64	10.13	94.44	18.71	56.02	13.25	3.34	73.39
DeepSeek-OCR [32]	arXiv 2025	-208.84	37.65	70.68	53.45	73.03	46.85	23.38	53.87
FireRed-OCR [34]	arXiv 2026	-244.73	61.03	46.42	68.48	79.03	65.26	43.39	26.67
dots.ocr [33]	arXiv 2025	-35.40	63.41	43.44	70.36	79.06	67.53	46.47	28.89
Qianfan-OCR [35]	arXiv 2026	4.58	68.87	37.57	75.31	80.72	73.16	51.40	26.86
HunyuanOCR [15]	arXiv 2025	48.87	89.62	12.12	89.91	<u>91.27</u>	89.18	79.91	<u>7.53</u>
MonkeyOCR-1.2B [37]	arXiv 2025	-422.25	19.90	87.49	30.15	49.18	25.78	7.12	59.57
MonkeyOCR-3B [37]	arXiv 2025	-169.16	37.57	70.34	48.86	58.40	45.30	18.51	57.85
MinerU2.5 [38]	arXiv 2025	24.83	35.24	68.57	49.89	55.33	48.82	23.13	57.12
PaddleOCR-VL-1.5 [36]	arXiv 2026	40.26	64.62	40.50	73.03	79.12	71.04	50.74	25.16
MinerU2.5-Pro [23]	arXiv 2026	38.48	60.20	45.75	69.96	70.52	70.79	45.39	35.39
GLM-OCR [16]	arXiv 2026	46.73	55.73	48.76	66.40	66.88	68.38	41.86	34.06
PaddleOCR-VL-1.6 [39]	arXiv 2026	53.59	74.00	29.93	78.90	81.39	78.08	59.72	16.91
TongGuOCR	This work	**93.76**	**94.46**	**6.15**	**94.44**	**94.79**	**94.14**	**88.92**	**3.49**

Note. Negative AR values occur when insertion errors are sufficiently severe that $S + D + I > N$, indicating over-generation rather than a computational error; see Section 4.2.

metrics, BLEU score, and RO-ED. The reductions in NED from 10.43 to 6.15 and in RO-ED from 7.53 to 3.49 indicate better preservation of both content and reading order. These gains show that the proposed framework is especially effective on historical pages where layout complexity, difficult character forms, and nontrivial reading paths jointly affect transcription quality.

Qualitative analysis.

Figure 4 provides a qualitative comparison from the complementary perspectives of reading-path prediction and text transcription. In the reading-path visualization, the compared systems exhibit missed lines, interrupted paths, or out-of-order transitions. GPT-5.5 and Qwen3.7-Plus recover portions of the page content but fail to establish a reliable traversal sequence across the dense table-like layout, while HunyuanOCR, PaddleOCR-VL-1.5, and GLM-OCR also exhibit omissions or inconsistent transitions. Most of

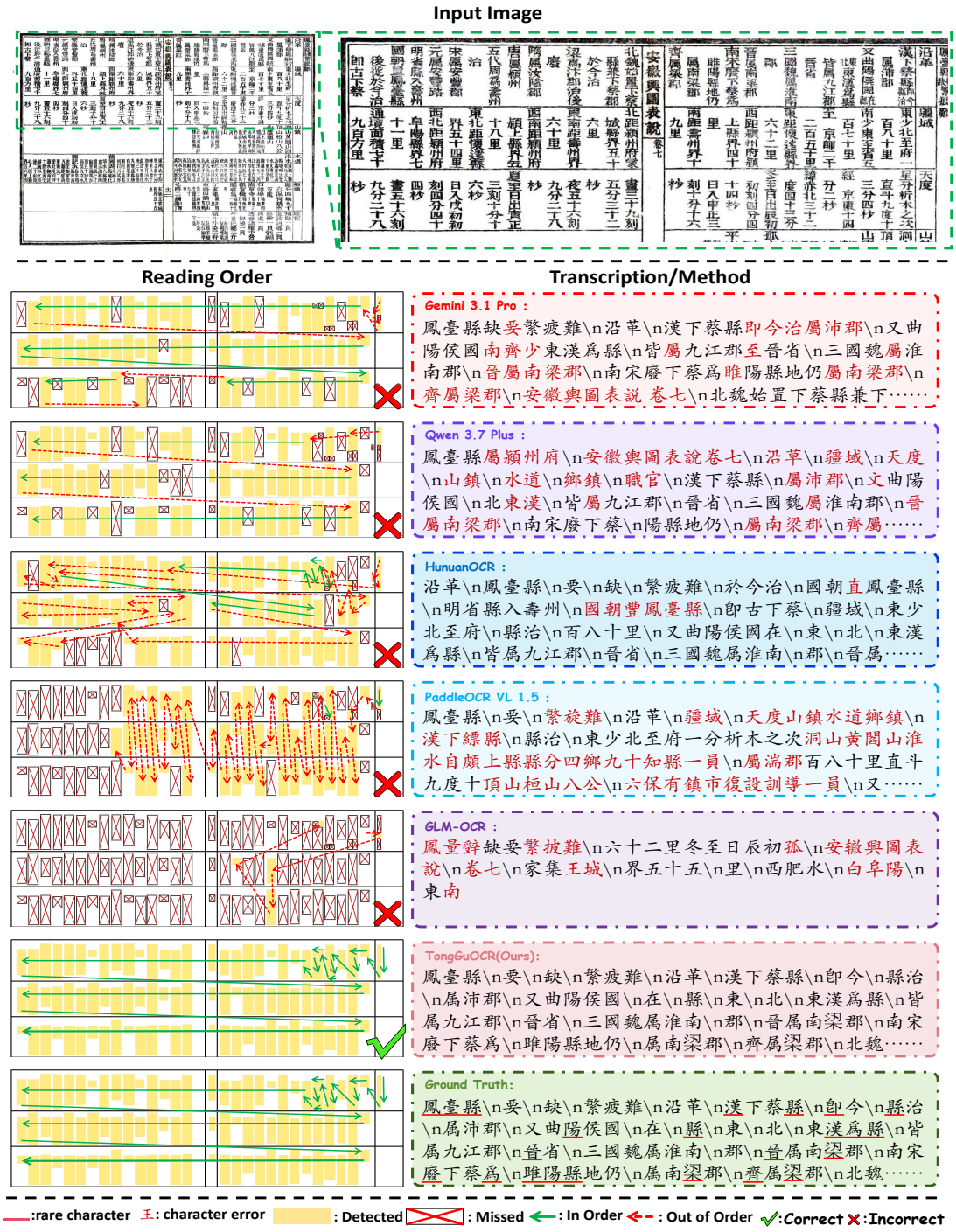

Fig. 4 Qualitative comparison on a challenging page from M^{5}HisDoc, which features a dense table-like layout and nontrivial reading-order transitions. The upper panel shows the input page and the enlarged target region. The lower-left panel visualizes the predicted reading paths, while the lower-right panel presents the corresponding transcriptions. Yellow boxes denote detected text lines, and crossed boxes indicate missed lines. Green and red arrows represent correct and incorrect reading-order transitions, respectively. Green checks and red crosses indicate whether the overall reading path is correct, while red underlining highlights rare or uncommon characters. Red characters, on the other hand, mark transcription errors. In this example, TongGuOCR recovers all target text lines, follows the natural reading sequence of the historical page, and accurately transcribes the corresponding content.

these systems are primarily optimized for modern document OCR and parsing. Although they retain some capability on historical materials, their predictions are less reliable on dense vertical layouts and nontrivial line transitions. In contrast, TongGuOCR recovers all target text lines and follows the natural reading sequence of the page.

The transcription panels show the consequences of the path errors described above. Missed text lines lead to omitted content, incorrect transitions place text segments in an unnatural sequence, and recognition errors alter individual characters. In contrast, TongGuOCR recovers every target line, preserves the natural reading sequence of the historical page, and accurately transcribes the corresponding content. This result provides a concrete illustration of the factors reflected in AR, NED, BLEU, and RO-ED: complete text recovery, accurate character recognition, and coherent line sequencing.

Overall, the results support the design of TongGuOCR, in which Layout-Aware Preprocessing provides structured and OCR-friendly recognition inputs, while Token-Augmented Recognition improves rare-character representation and provides spatial guidance during transcription through line-to-line transition modeling.

4.5 Ablation Studies

We organize the ablations around the two-module architecture and evaluate the staged SFT strategy separately. For Layout-Aware Preprocessing, we examine input-granularity alignment between SFT and inference and analyze the sensitivity to the segmentation penalty weights. For Token-Augmented Recognition, we ablate character-level vocabulary expansion and line-to-line transition modeling, followed by a detailed analysis of added and non-added character subsets. We then evaluate the contribution of each SFT stage.

All results are computed on the full official M^{5}HisDoc test split. We use this benchmark for ablation because its diverse layouts, visual styles, and historical character forms provide a more discriminative test than MTHv2 [26], on which the strongest baselines are already close to saturation (Table 3). The module ablations exclude HisDoc1B because repeating full-parameter SFT on this large dataset for every variant would be prohibitively expensive. These variants are trained on the official training splits of MTHv2 and M^{5}HisDoc, with checkpoint selection on the official M^{5}HisDoc validation split. MTHv2 is used only for training in these ablations.

The staged SFT analysis follows the main training procedure and includes HisDoc1B in Stage 1. Its absolute scores should therefore not be compared with those in the module ablations, which use a different training-data configuration. Comparisons within each table remain controlled under the same protocol.

4.5.1 Effect of Layout-Aware Preprocessing

Effectiveness of Recognition Blocks. This experiment evaluates whether recognition blocks improve OCR performance over full-page inputs and whether they should be used consistently during both SFT and inference. Page-level SFT uses full-page samples, whereas block-level SFT uses recognition-block crops. Inference is likewise performed with either full pages or recognition blocks. Page/Page and Block/Block therefore use matched input granularities, while Page/Block and Block/Page introduce a mismatch. All four configurations retain character-level vocabulary expansion and line-to-line transition modeling.

Table 5 shows that Block/Block performs best on every metric, achieving 89.69 AR, 90.88 CR, 82.30 BLEU, and 4.93 RO-ED. Compared with Page/Page, it improves AR by 6.19 points and BLEU by 8.93 points, while reducing NED from 14.00 to 9.79. These results suggest that recognition blocks are most effective when used consistently during both SFT and inference. Because this comparison changes the input granularity at both stages, it supports the complete recognition-block interface rather than an isolated test-time preprocessing effect.

The remaining configurations perform less consistently for different reasons. Page/Block introduces recognition blocks only during inference, even though the model is trained with full-page images. Although CR increases from 86.66 to 87.19 and BLEU from 73.37 to 74.10, AR decreases from 83.50 to 75.43, showing that this test-time change does not provide a consistent benefit. Block/Page creates the opposite mismatch and performs worst, with 68.47 AR and 26.60 RO-ED. Its F1 remains relatively high at

Table 5 Effect of input-granularity alignment between SFT and inference on the M^{5}HisDoc test split. "Page" and "Block" denote full-page images and recognition-block crops, respectively.

SFT	Infer.	M^{5}HisDoc							
		AR↑	CR↑	NED↓	F1↑	P↑	R↑	BLEU↑	RO-ED↓
Page	Page	83.50	86.66	14.00	84.86	85.98	84.03	73.37	5.24
Page	Block	75.43	87.19	13.93	85.55	86.15	85.08	74.10	5.26
Block	Page	68.47	74.11	29.83	86.34	88.10	85.44	72.61	26.60
Block	Block	**89.69**	**90.88**	**9.79**	**90.17**	**90.32**	**90.07**	**82.30**	**4.93**

Table 6 Sensitivity analysis of penalty weights in Recognition Block Construction on the M^{5}HisDoc test split. Each setting is represented as $(\lambda_{\text{blk}}, \lambda_{\text{adj}}, \lambda_{\text{page}})$, where the three coefficients weight the block-level, adjacent-block, and page-level penalties, respectively. † denotes the default setting used in our main ablation experiments.

Setting	M^{5}HisDoc							
	AR↑	CR↑	NED↓	F1↑	P↑	R↑	BLEU↑	RO-ED↓
(1, 1, 1)	85.44	90.06	10.75	89.18	89.31	89.11	80.46	5.29
(2, 1, 1)	85.94	89.92	11.06	89.33	89.85	89.02	80.47	5.38
(1, 2, 1)	86.75	90.17	10.66	89.55	89.92	89.29	80.87	5.10
(4, 2, 1)	87.96	89.89	10.82	89.16	89.51	88.92	80.50	5.07
(2, 4, 1)	86.93	89.74	10.95	89.09	89.62	88.78	80.35	5.48
(4, 4, 1)	84.57	88.91	11.87	87.95	88.46	87.64	78.54	5.34
(2, 2, 1)†	**89.69**	**90.88**	**9.79**	**90.17**	**90.32**	**90.07**	**82.30**	**4.93**

86.34, but this set-based metric does not capture the severe errors in complete transcription and reading order. Page/Page avoids a granularity mismatch but still underperforms Block/Block, a pattern consistent with the loss of character detail and cross-region interference associated with full-page inputs. Overall, the results support the consistent use of recognition blocks during both SFT and inference.

Penalty-weight sensitivity. Based on the aligned Block/Block configuration, Table 6 evaluates the relative weights of the three penalty groups in Recognition Block Construction. Since multiplying all three weights by the same positive constant does not change the optimal segmentation, we fix $\lambda_{\text{page}} = 1$ as the reference and vary λ_{blk} and λ_{adj}. The experiments therefore compare different relative weight ratios rather than their absolute magnitudes.

The default ratio (2,2,1) performs best on every metric, achieving 89.69 AR, 9.79 NED, and 4.93 RO-ED. The ratios (4,2,1) and (1,2,1) remain competitive on different metrics, but neither matches the default setting. Increasing both local penalty weights to (4,4,1) reduces AR to 84.57 and raises NED to 11.87. This decline suggests that excessive emphasis on block-level and adjacent-block constraints can overconstrain the segmentation. The default ratio therefore provides the best balance among block quality, adjacent-block compatibility, and page-level segmentation complexity.

4.5.2 Effect of Token-Augmented Recognition

This ablation evaluates the two token-level designs introduced in Sections 3.2.1 and 3.2.2: character-level vocabulary expansion and line-to-line transition modeling. All variants use recognition-block inputs during both SFT and inference, following the aligned setting established in the previous ablation, and only these two token-level designs are changed.

Effects of Vocabulary Expansion and Transition Modeling. Table 7 shows that both designs improve the baseline and provide complementary benefits. Vocabulary expansion raises

Table 7 Ablation of character-level vocabulary expansion and line-to-line transition modeling on the M^{5}HisDoc test split. "Vocab." denotes character-level vocabulary expansion, and "Trans." denotes line-to-line transition modeling.

Vocab.	Trans.	M^{5}HisDoc							
		AR↑	CR↑	NED↓	F1↑	P↑	R↑	BLEU↑	RO-ED↓
–	–	80.89	89.67	11.23	88.92	89.12	89.05	80.44	5.56
✓	–	86.12	90.13	10.65	89.47	90.01	89.23	81.10	5.23
–	✓	87.89	90.12	10.65	89.18	89.14	89.36	80.94	5.24
✓	✓	**89.69**	**90.88**	**9.79**	**90.17**	**90.32**	**90.07**	**82.30**	**4.93**

Table 8 Occurrence-level character alignment analysis of token-augmented recognition on the M^{5}HisDoc test split. Characters are divided into Added chars and Non-added chars according to whether they belong to the newly introduced single-character tokens. "Vocab." denotes character-level vocabulary expansion, and "Trans." denotes line-to-line transition modeling. FP and FN denote over-generated and missed character occurrences after full-text character-level alignment, respectively.

Vocab.	Trans.	Added chars					Non-added chars		
		P↑	R↑	F1↑	FP↓	FN↓	P↑	R↑	F1↑
–	–	75.17	78.73	76.91	33270	27219	79.26	91.21	84.82
✓	–	84.39	78.43	81.30	18567	27602	84.55	92.17	88.19
–	✓	81.66	79.21	80.42	22758	26605	88.56	92.00	90.25
✓	✓	**85.43**	**80.42**	**82.85**	**17547**	**25057**	**91.42**	**92.80**	**92.11**

AR from 80.89 to 86.12 and reduces NED from 11.23 to 10.65, indicating more stable character generation after reducing token fragmentation. Transition modeling raises AR to 87.89, which suggests that explicit spatial cues help the decoder move from a completed line to the next relevant region. The two single-component variants achieve comparable RO-ED scores. Their combination performs best on every metric, reaching 89.69 AR, 9.79 NED, 82.30 BLEU, and 4.93 RO-ED.

The transition-only variant obtains an RO-ED of 5.24, which is slightly higher than the 5.23 achieved by vocabulary expansion alone. This small irregularity may arise because RO-ED is computed after predicted lines are matched to reference lines according to textual similarity. Recognition errors can therefore affect line matching and the resulting order score. The 0.01 difference does not contradict the overall benefit of transition modeling, which improves AR from 80.89 to 87.89.

Performance on Added and Non-Added Characters. Table 8 shows how the two designs affect different character subsets. Added chars are occurrences represented by the new single-character tokens, while Non-added chars are the remaining normalized occurrences. The baseline obtains F1 scores of 76.91 and 84.82 on these subsets, respectively, confirming that the added characters are more difficult.

Vocabulary expansion primarily benefits the added-character subset. Without transition modeling, it raises precision from 75.17 to 84.39 and F1 from 76.91 to 81.30. Recall decreases slightly from 78.73 to 78.43, but false positives fall from 33,270 to 18,567. The expanded vocabulary therefore makes predictions more precise with little loss in coverage.

Transition modeling provides a different benefit. Without vocabulary expansion, it raises F1 from 84.82 to 90.25 for non-added characters and from 76.91 to 80.42 for added characters. This result suggests that better sequence organization reduces alignment errors caused by complex layouts. Combining both designs produces the best F1 on each subset, with scores of 82.85 for added characters and 92.11 for non-added characters. Vocabulary expansion improves difficult character

Table 9 Effect of staged SFT on the M^{5}HisDoc test split. Stage 1 denotes coarse domain adaptation, Stage 2 denotes high-quality page-level refinement, and Stage 3 denotes block-level inference alignment. The reported rows correspond to the final Stage 1 checkpoint at epoch 3 and the validation-selected Stage 2 and Stage 3 checkpoints at epochs 5 and 3, respectively.

Stage 1	Stage 2	Stage 3	M^{5}HisDoc							
			AR↑	CR↑	NED↓	F1↑	P↑	R↑	BLEU↑	RO-ED↓
✓	–	–	90.29	91.55	9.57	93.08	92.95	93.28	84.32	6.42
✓	✓	–	90.03	93.56	7.47	93.75	94.18	93.42	86.93	4.07
✓	✓	✓	**93.76**	**94.46**	**6.15**	**94.44**	**94.79**	**94.14**	**88.92**	**3.49**

representations, while transition modeling stabilizes the output sequence.

4.5.3 Effect of Staged Supervised Fine-Tuning

This ablation examines whether the three SFT stages fulfill their intended roles in adapting domain knowledge, improving supervision quality, and aligning input granularity. The three rows correspond to coarse domain adaptation, high-quality page-level refinement, and block-level inference alignment. All models use recognition-block inputs during evaluation.

Table 9 shows that Stage 1 establishes a strong historical-document OCR foundation through large-scale domain adaptation. Stage 2 then uses human-annotated page-level data for high-quality refinement, improving CR from 91.55 to 93.56, reducing NED from 9.57 to 7.47, increasing BLEU from 84.32 to 86.93, and reducing RO-ED from 6.42 to 4.07. These improvements are consistent with its intended role in refining character recognition and overall sequence fidelity. AR decreases only slightly from 90.29 to 90.03; since AR additionally penalizes insertions while all other reported metrics improve, this minor exception reflects increased insertion errors rather than an overall performance degradation. These additional insertions may partly arise from the remaining granularity mismatch between page-level training and recognition-block inference.

Stage 3 aligns training with inference by using recognition blocks generated through the same preprocessing procedure used at test time. It improves every metric, raising AR from 90.03 to 93.76 and CR from 93.56 to 94.46, while reducing NED from 7.47 to 6.15 and RO-ED from 4.07 to 3.49. AR improves more than CR, and the gap between CR and AR decreases from 3.53 to 0.70. These changes indicate fewer insertion errors and support input-granularity alignment. Although Stages 2 and 3 are scheduled for 10 and 15 epochs, their validation-selected checkpoints occur at epochs 5 and 3, respectively. Continuing either stage produces no better checkpoint. The gains therefore reflect the intended changes in supervision quality and input granularity rather than longer training alone.

5 Discussion

The results show that reliable OCR for Chinese historical documents requires coordinated treatment of layout, character representation, and reading order. Recognition blocks preserve character visibility while retaining local context. Character-level vocabulary expansion creates a more direct correspondence between visual glyphs and output tokens for rare characters. Transition tokens provide spatial supervision at line boundaries and help the decoder maintain a coherent reading path. Together, these designs achieve 93.76 AR and 3.49 RO-ED on M^{5}HisDoc, reflecting strong character recognition and reading-order preservation.

The strong performance of TongGuOCR across the evaluated benchmarks further demonstrates the effectiveness of this design. Compared with existing approaches, TongGuOCR achieves more accurate and stable transcription for Chinese historical documents, especially in scenarios where layout complexity and rare characters substantially affect recognition quality. These results suggest that explicitly modeling layout structure and historical character distributions is important for

improving OCR performance on historical materials. Beyond benchmark accuracy, TongGuOCR also has practical value for digital humanities and cultural heritage preservation, where reliable transcription can support full-text retrieval, textual collation, named-entity extraction, knowledge organization, and large-scale access to image-based historical collections.

TongGuOCR still has two limitations. First, the expanded vocabulary is derived from the training corpus and does not cover all historical characters, including those absent from the training data or not yet deciphered. Second, the evaluated datasets do not represent the full diversity of Chinese historical materials. Future work will investigate more flexible representations for unseen characters and evaluate the framework on more diverse historical materials.

6 Conclusion

In this paper, we present TongGuOCR, a new OCR-oriented MLLM framework designed for the distinctive challenges of Chinese historical documents. Faithful OCR of Chinese historical documents requires coordinated modeling of recognition granularity, character representation, and cross-line structure. Our results show that recognition blocks are most effective when training and inference use the same granularity, and that direct character-level representations and explicit line-boundary supervision provide complementary benefits for rare-character fidelity and multi-line sequence organization. These findings establish a broader design principle: layout-guided inputs and structured transcription targets should be designed jointly rather than as isolated components. TongGuOCR validates this principle across representative benchmarks and provides a stronger foundation for retrieval, textual collation, and computational study of Chinese historical collections.

Code availability

The source code and model checkpoints will be made publicly available upon acceptance and publication of this paper at https://github.com/jzzh2004/TongGuOCR.

Acknowledgements

This research is supported in part by the National Natural Science Foundation of China (Grant No. 62476093), the Natural Science Foundation of Guangdong Province (Grant No. 2026A1515012038), the China Postdoctoral Science Foundation (Grant No. 2026M791625), and the Postdoctoral Fellowship Program (Grade B) of the China Postdoctoral Science Foundation (Grant No. GZB20260386).

References

[1] Wang, Z., Zhang, C., Lang, Q., Jin, L. & Qi, H. HUNet: hierarchical universal network for multi-type ancient Chinese character recognition. *npj Heritage Science* **13**, 281 (2025).

[2] Guo, H., Zhu, D., Tong, L., Zhao, J. & Jiang, A. GLADUnet: global-local attention and dynamic upsampling for ancient manuscript denoising and binarization. *npj Heritage Science* **13**, 662 (2025).

[3] Li, X., Huang, Y., She, S. & Xie, J. Restoration of Liye Qin slips characters using generative adversarial network with effective connected component constraint. *npj Heritage Science* **14**, 194 (2026).

[4] Wang, K., Wu, T., Shi, Y. & Chen, C. AttGraph disentangling confusable ancient Chinese characters via component-correlation synergy. *npj Heritage Science* **14**, 7 (2026).

[5] Lombardi, F. & Marinai, S. Deep Learning for Historical Document Analysis and Recognition—A Survey. *Journal of Imaging* **6** (2020).

[6] Sturgeon, D. Digitizing Premodern Text with the Chinese Text Project. *Journal of Chinese History* **4**, 486–498 (2020).

[7] Sturgeon, D. Chinese Text Project: A dynamic digital library of premodern Chinese. *Digital Scholarship in the Humanities* **36**, i101–i112 (2021).

[8] De Weerdt, H. Creating, Linking, and Analyzing Chinese and Korean Datasets: Digital Text Annotation in MARKUS and COMPARATIVUS. *Journal of Chinese History* **4**, 519–527 (2020).

[9] Xu, Q., Liu, Y., Wang, D. & Huang, S. Automatic recognition of cross-language classic entities based on large language models. *npj Heritage Science* **13**, 59 (2025).

[10] Zhao, Z., Sun, G., Liu, C. & Wang, D. Research on machine translation of ancient books in the era of large language model. *npj Heritage Science* **13**, 122 (2025).

[11] Liu, J. *et al.* Multi-dimensional intelligent reorganization and utilization of knowledge in 'Biographies of Chinese Thinkers'. *npj Heritage Science* **13**, 135 (2025).

[12] Cao, J. *et al.* Tonggu: Mastering classical chinese understanding with knowledge-grounded large language models. *EMNLP 2024* (2024).

[13] Sturgeon, D. Large-scale Optical Character Recognition of Pre-modern Chinese Texts. *International Journal of Buddhist Thought and Culture* **28**, 11–44 (2018).

[14] Ma, H., Huang, H. & Liu, C. *Reading between the lines: Image-based order detection in OCR for chinese historical documents. Proceedings of the AAAI Conference on Artificial Intelligence*, 23808–23810 (AAAI Press, 2024).

[15] Team, H. V. *et al.* HunyuanOCR Technical Report (2025).

[16] Duan, S. *et al.* GLM-OCR Technical Report (2026). arXiv:2603.10910.

[17] Liu, C. *et al.* A robust and efficient algorithm for Chinese historical document analysis and recognition. *National Science Review* **10**, nwad115 (2023).

[18] Ma, W. *et al.* Joint Layout Analysis, Character Detection and Recognition for Historical Document Digitization. *ICFHR 2020* (2020).

[19] Wang, H.-A. & Liu, P.-T. *Towards a higher accuracy of optical character recognition of chinese rare books in making use of text model. Proceedings of the 3rd International Conference on Digital Access to Textual Cultural Heritage*, DATeCH2019, 15–18 (Association for Computing Machinery, New York, NY, USA, 2019).

[20] Bai, S. *et al.* Qwen3-VL Technical Report. *arXiv preprint arXiv:2511.21631* (2025).

[21] Wang, W. *et al.* InternVL3.5: Advancing Open-Source Multimodal Models in Versatility, Reasoning, and Efficiency. *arXiv preprint arXiv:2508.18265* (2025).

[22] Wei, H., Sun, Y. & Li, Y. Deepseek-ocr 2: Visual causal flow. *arXiv preprint arXiv:2601.20552* (2026).

[23] Wang, B. *et al.* MinerU2.5-Pro: Pushing the Limits of Data-Centric Document Parsing at Scale. *arXiv preprint arXiv:2604.04771* (2026).

[24] Sihang, W., Jiapeng, W., Weihong, M. & Lianwen, J. Precise detection of chinese characters in historical documents with deep reinforcement learning. *Pattern Recognition* **107**, 107503 (2020).

[25] Yang, H. *et al.* Dense and tight detection of chinese characters in historical documents: Datasets and a recognition guided detector. *IEEE Access* **6**, 30174–30183 (2018).

[26] Shi, Y. *et al.* *M5hisdoc: A large-scale multi-style chinese historical document analysis benchmark. Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track* (2023).

[27] Shi, Y., Peng, D., Zhang, Y., Cao, J. & Jin, L. A large-scale dataset for Chinese historical document recognition and analysis. *Scientific Data* **12**, 169 (2025).

[28] Team, Q. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804* (2026).

[29] Qwen Team. Qwen3.7-Plus: Multimodal Agent Intelligence (2026). URL https://qwen.ai/blog?id=qwen3.7-plus.

[30] Google. Gemini 3.1 pro preview. https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview (2026).

[31] OpenAI. Introducing gpt-5.5. https://openai.com/index/introducing-gpt-5-5/ (2026).

[32] Wei, H., Sun, Y. & Li, Y. DeepSeek-OCR: Contexts Optical Compression. *arXiv preprint arXiv:2510.18234* (2025).

[33] Li, Y., Yang, G., Liu, H., Wang, B. & Zhang, C. dots.ocr: Multilingual Document Layout Parsing in a Single Vision-Language Model (2025). arXiv:2512.02498.

[34] Super Intelligence Team, X. I. Firered-ocr technical report (2026).

[35] Dong, D. *et al.* Qianfan-OCR: A Unified End-to-End Model for Document Intelligence (2026). arXiv:2603.13398.

[36] Cui, C. *et al.* PaddleOCR-VL-1.5: Towards a Multi-Task 0.9B VLM for Robust In-the-Wild Document Parsing (2026). arXiv:2601.21957.

[37] Li, Z. *et al.* MonkeyOCR: Document Parsing with a Structure-Recognition-Relation Triplet Paradigm (2025). arXiv:2506.05218.

[38] Niu, J. *et al.* Mineru2.5: A decoupled vision-language model for efficient high-resolution document parsing. *arXiv preprint arXiv:2509.22186* (2025).

[39] Zhang, Z. *et al.* Paddleocr-vl-1.6: Expanding the frontier of document parsing with under-optimized region refinement and progressive post-training (2026). arXiv:2606.03264.

[40] Zhang, P., Zhang, J., Cao, J., Li, H. & Jin, L. Smaller But Better: Unifying Layout Generation with Smaller Large Language Models. *International Journal of Computer Vision (IJCV)* **133**, 3891–3917 (2025).

[41] Ye, J. *et al.* Ureader: Universal ocr-free visually-situated language understanding with multimodal large language model (2023). arXiv:2310.05126.

[42] Liu, Y. *et al.* TextMonkey: An ocr-free large multimodal model for understanding document (2024). URL https://arxiv.org/abs/2403.04473. arXiv:2403.04473.

[43] Zhang, H., Whittaker, E. & Kitagishi, I. Extending TrOCR for Text Localization-Free OCR of Full-Page Scanned Receipt Images. *arXiv e-prints* arXiv:2212.05525 (2022).

[44] Ke, W. *et al.* Large language models in document intelligence: A comprehensive survey, recent advances, challenges, and future trends. *ACM Transactions on Information Systems* **44**, 1–64 (2025).

[45] Cui, C. *et al.* PaddleOCR-VL: Boosting Multilingual Document Parsing via a 0.9B Ultra-Compact Vision-Language Model. *arXiv e-prints* arXiv:2510.14528 (2025).

[46] Zhu, Y. *et al.* Unit: Unifying image and text recognition in one vision encoder. *Advances in Neural Information Processing Systems* **37**, 122185–122205 (2024).

[47] Kudo, T. & Richardson, J. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66–71 (Association for Computational Linguistics, Brussels, Belgium, 2018).

[48] Liu, Y. *et al. MCS-bench: A comprehensive benchmark for evaluating multimodal large language models in Chinese classical studies. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 10435–10492 (Association for Computational Linguistics, 2025).

[49] Ouyang, L. *et al.* Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations (2024).

arXiv:2412.07626.

[50] Zheng, Y., Zhang, R., Zhang, J., Ye, Y. & Luo, Z. *LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, 400–410 (Association for Computational Linguistics, 2024).

[51] Loshchilov, I. & Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[52] Huang, Y., Jin, L. & Peng, D. *Zero-shot chinese text recognition via matching class embedding. Document Analysis and Recognition – ICDAR 2021*, 127–141 (Springer International Publishing, Cham, 2021).

[53] Zhang, J., Du, J. & Dai, L. Radical analysis network for learning hierarchies of chinese characters. *Pattern Recognition* **103**, 107305 (2020).

[54] Shi, B., Bai, X. & Yao, C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence* **39**, 2298–2304 (2016).

[55] Lai, X. *et al.* MiniMax Sparse Attention (2026). arXiv:2606.13392.

[56] Moonshot AI. Kimi-k2.6. Hugging Face model card (2026). URL https://huggingface.co/moonshotai/Kimi-K2.6.

[57] Shi, B. *et al.* Aster: An attentional scene text recognizer with flexible rectification. *IEEE transactions on pattern analysis and machine intelligence* **41**, 2035–2048 (2018).

[58] Sheng, F., Chen, Z. & Xu, B. NRTR: A no-recurrence sequence-to-sequence model for scene text recognition. *CoRR* **abs/1806.00926** (2018).

[59] Yue, X., Kuang, Z., Lin, C., Sun, H. & Zhang, W. *RobustScanner: Dynamically Enhancing Positional Clues for Robust Text Recognition. Computer Vision – ECCV 2020*, 135–151 (Springer International Publishing, Cham, 2020).

[60] Peng, D. *et al.* Recognition of handwritten chinese text by segmentation: a segment-annotation-free approach. *IEEE Transactions on Multimedia* **25**, 2368–2381 (2022).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- TongGuOCRsupplementary.pdf