STRUCTURE BEFORE GENERATION: INVARIANT GRAPH READOUT WITH PROTECTED COMPLETION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models often reconstruct document structure by generating serialized graphs, coupling structural decisions to output ordering, identifier copying, and syntax. Direct graph predictors remove this serialization step but do not by themselves specify how an existing prediction can be revised or extended without disturbing already-correct structure. We introduce Invariant Graph Readout, a non-autoregressive framework that combines direct graph prediction from frozen visual representations of a pretrained vision-language model with state-preserving correction and protected completion. A trainable query head predicts typed nodes, bounding boxes, and relations, while operation-specific preservation contracts define which attributes each graph-edit operation may modify and which state it must preserve. Complementary proposals, anchor-preserving selection, and endpoint-consistent attachment recover missing structure without rewriting protected objects. On 600 held-out OracleGraph pages, the complete system achieves a Graph F1 of 0.3988 ± 0.0023 versus 0.3872 ± 0.0072 for Strict JSON Generation (mean ± standard deviation over three training seeds). On a 100-page subset at batch size one on an A100 GPU, it reduces mean end-to-end latency by 22.3× relative to greedy Strict JSON Generation (3.60 vs. 80.14 s per page), whose outputs average approximately 1,157 tokens per page. Under supervised target-domain training on GraphDoc, mean Graph F1 on 1,000 held-out pages improves from 0.0443 for a global-and-local graph readout baseline to 0.1069 across three independently trained pipelines. These results indicate that preservation-aware graph editing built on direct prediction is a practical alternative to autoregressive structural generation, and that contract-guided completion can be reused across document domains under supervised training.

1 INTRODUCTION

Generative vision-language models commonly represent document structure as serialized text. Yet the output interface of a pretrained model need not dictate the representation of the task it solves. When the required output is a typed graph rather than a transcription, serialization introduces dependencies on output order, repeated identifiers for relation endpoints, and syntactic validity, and its cost grows with the number of generated structural tokens. Direct set and graph predictors (Carion et al., 2020; Chen et al., 2025) already avoid this step, showing that structured outputs need not be generated as sequences. Direct prediction, however, addresses initial graph construction: it specifies what structure to produce from an input, but not how an existing prediction can subsequently be revised or extended without disturbing structure that is already correct. We therefore ask how a graph-native interface can support targeted correction and recovery of missing structure while explicitly preserving designated parts of the current prediction.

Correction and completion address complementary failure modes. Geometric updates can alter representations also used for node classification, and node deletion can remove endpoints required by otherwise valid relations. Both errors arise when a refinement operation modifies graph state that it was not intended to change. Correction also has an inherent limit: revising retained predictions cannot recover an entity absent from the current candidate set. A document-graph interface therefore requires more than direct prediction. It must distinguish operations that revise existing structure within explicit preservation boundaries from operations that introduce missing nodes and relations without rewriting accepted objects.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We introduce **Invariant Graph Readout**, a non-autoregressive framework that provides such an interface by combining direct graph prediction from frozen visual representations of a pretrained vision-language model with state-preserving correction and protected completion. Each component targets one of the failure modes above. Within the trainable query head, node classification reads the query state before geometric refinement, so subsequent box updates cannot alter the features that determine node types. At the graph level, operation-specific preservation contracts declare which attributes each graph-edit operation may modify and which state it must preserve; for example, node cleanup may not remove nodes that serve as endpoints of retained relations. Protected completion then recovers structure absent from the candidate set through complementary proposals, anchor-preserving selection, and endpoint-consistent attachment, without rewriting protected objects. The contract-guided completion interface and its core preservation requirements are shared across domains, whereas starting-graph procedures, proposal generators, learned models, and calibration rules are domain-specific.

We evaluate the framework in two domains: OracleGraph (Sun et al., 2026), a procedurally generated dataset of historical-style document pages, and GraphDoc (Chen et al., 2025), a dataset of document pages annotated with layout elements and their relations. Each instantiation is trained with supervision in its own domain, which lets us test whether the shared interface carries over when the domain-specific components change. Our contributions are threefold:

- **A graph-native correction-and-completion framework.** We organize document-graph recovery around explicit nodes, stable identities, and typed relations, separating bounded edits to existing predictions from additions that recover missing structure.
- **Operation-specific preservation contracts.** We define explicit edit boundaries that distinguish writable graph attributes from protected state for each operation, providing a common abstraction for correction and completion.
- **Quality, efficiency, and component-level evidence.** We compare graph quality and end-to-end latency with Strict JSON Generation on OracleGraph and evaluate supervised reuse on GraphDoc across three independently trained pipelines. Matched-count controls distinguish overall system gains from the conditional effect of residual candidate reranking, while stage-wise analyses describe pipeline progression.

2 Related Work

Generative document understanding. OCR-free document models commonly produce structure through the language outputs of generative vision-language models (Wang et al., 2024c; Hu et al., 2025; Wang et al., 2024a). Work on high-resolution perception, including frequency-domain encoding (Feng et al., 2024), efficient vision encoders (Vasu et al., 2025), visual-token compression (Hu et al., 2025), and mask-based spatial alignment (Wang et al., 2025b), improves how visual inputs are processed, while parallel-token prediction (Li et al., 2026; Cao et al., 2026) and discrete diffusion (Li et al., 2025; Wang et al., 2025a) modify the generation process itself. These directions improve the efficiency or capacity of generative document models but retain a token-space output interface, in which relations are expressed through copied identifiers and structural validity depends on syntax. They are complementary to our setting, where the target is a typed document graph with localized nodes and explicit relation endpoints. Sun et al. (2026) introduced OracleGraph for benchmarking multimodal structured generation and PRISM for optimizing generative models on this task. Invariant Graph Readout instead asks whether document structure can be constructed through a non-autoregressive graph interface that supports not only direct prediction, but also controlled correction and recovery of missing structure.

Direct structural prediction. Query-based set predictors and image-to-graph models already show that structured outputs need not be serialized autoregressively (Carion et al., 2020). In documents, layout analysis (Chen et al., 2026), form information extraction (Lee et al., 2022; 2023), hierarchical structure reconstruction (Ma et al., 2023; Wang et al., 2024b; Shin et al., 2025), and document-graph prediction (Chen et al., 2025) address related output requirements. We therefore do not claim non-autoregressive graph prediction itself as a contribution. Instead, we start from a limitation that remains after direct prediction: a direct predictor specifies what structure to produce from an input, but not how an existing prediction may be revised or extended without disturbing

structure that is already correct. As discussed in Section 1, correction can interfere with retained state, and correction alone cannot recover entities absent from the candidate set. Our contribution is the graph-edit interface that follows direct prediction: how correction and completion are composed while explicitly designated graph state is preserved.

Structured refinement and adaptation. Generated structures can violate topological constraints (Yang et al., 2025), and document extraction systems degrade under real-world noise (Ouyang et al., 2025; Chen et al., 2024). Approximate graph edit distance (Pellizzoni et al., 2026; Guo et al., 2026), flow-matching graph generation (Hu et al., 2026), and data-driven structure refinement (Yun et al., 2025) revise or regenerate structure toward a target, typically specifying what the refined output should be rather than which parts of the current prediction each update must leave unchanged. Invariant Graph Readout makes this retention explicit through operation-specific preservation contracts: each graph-edit operation declares which attributes it may modify and which state it must preserve. Under these contracts, geometric correction preserves node types, identities, and relations; node cleanup preserves the endpoints of retained edges; and protected completion adds complementary nodes and relations without rewriting accepted objects. Because the contracts are defined at the graph level, domain-specific proposal generators, learned models, and calibration rules can be exchanged beneath the same interface, as we do for OracleGraph and GraphDoc. Here, invariance refers to state preserved under a designated graph operation, not to invariance under image transformations or domain shifts.

3 METHOD

3.1 GRAPH READOUT AND EDIT CONTRACTS

Invariant Graph Readout unifies direct prediction, state-preserving correction, and protected completion as contract-governed construction of an explicit document graph. Its interface specifies not only what structure to predict, but also what subsequent operations are forbidden to change. Direct readout constructs the initial graph; correction revises retained predictions within declared edit boundaries; protected completion recovers missing structure without rewriting designated objects.

For an input image $\boldsymbol{x}$, the frozen visual encoder and merger of a pretrained vision-language model produce visual features $F_{\mathrm{v}}(\boldsymbol{x})$. A trainable query head predicts node types, boxes, and relations from these features. We represent a predicted graph as $G = (V, E)$, with

$$\begin{aligned} V &= \{(u_i, c_i, \boldsymbol{b}_i)\}_{i=1}^{n}, \\ U(V) &= \{u_i : (u_i, c_i, \boldsymbol{b}_i) \in V\}, \\ E &\subseteq U(V) \times \mathcal{R} \times U(V). \end{aligned} \tag{1}$$

Here, $u_i \in \mathcal{U}$ is a stable identity, $c_i \in \mathcal{C}$ is a node type, $\boldsymbol{b}_i \in \mathbb{R}^4$ is a box in image coordinates, and $\mathcal{R}$ is the set of relation types. Relation endpoints refer to stable identities rather than mutable array positions, so reindexing does not change which nodes an edge connects. The representation supports one or multiple relation labels per node pair, according to the domain ontology. We call the underlying non-autoregressive prediction interface *Direct Readout*. It constructs the graph without generating structural tokens; JSON export is deterministic serialization of an already constructed graph.

The central abstraction is an operation-specific preservation contract, not a fixed cascade of modules. Conditioned on the image, trained models, and domain configuration, an operation T declares writable attributes $\mathcal{W}_T$ and protected attributes $\mathcal{P}_T$. The protected objects and attributes are specified relative to the operation's input. If $\Pi_{\mathcal{P}_T}$ extracts this protected state, every admissible edit must satisfy

$$\Pi_{\mathcal{P}_T}(T(G)) = \Pi_{\mathcal{P}_T}(G), \qquad \forall G \in \mathcal{G}_T, \tag{2}$$

where $\mathcal{G}_T$ contains well-formed predicted graphs satisfying the operation's preconditions.

Equation 2 imposes a hard constraint on graph-state transitions. Implementations enforce it by restricting writes to the declared writable state, making the protected projection immutable under each admissible operation. Thus, box correction cannot alter protected node identities, types, or edges, and cleanup cannot delete endpoints of retained relations. These forms of off-target modification are

excluded by construction, rather than merely discouraged through an optimization objective. The resulting guarantee is exact preservation of designated predicted state, not ground-truth correctness or monotone F_1; the overall pipeline may still add or delete nodes.

Table 1: Preservation contracts in the readout head and graph-edit interface. SemGuard controls latent-state routing inside the head; the remaining rows describe graph-level operations or candidate generation that leaves the graph unchanged.

Component	Permitted change	Protected state
SemGuard (in-head)	Update geometric and relation representations	Classification reads the original query-decoder state
RelCal	Filter existing relations	Compiled nodes and their attributes
BoxFix	Correct existing boxes	Node identities, types, order, and edges
DupClean	Remove eligible isolated nodes	Retained-edge endpoints, surviving attributes, and edges
NodePropose	Generate candidate banks	Existing graph during proposal generation
AnchorSelect	Append selected candidates	Existing graph and accepted anchors
EdgeAttach	Append endpoint-consistent relations	Selected nodes and protected reference edges

Table 1 makes the preservation requirements explicit across correction and completion. The graph-state interface and core completion contracts are shared across domains, while starting-graph procedures, proposal generators, learned models, and calibration rules are domain-specific. SemGuard is the head-level counterpart of this separation; it is applied before graph compilation rather than as a graph edit.

3.2 State-preserving correction

SemGuard: separating semantic and geometric state. Inside the query head, classification reads the original decoder state $\boldsymbol{Z}^{(0)}$, while box-conditioned residual updates produce the state used for localization and relation prediction. This routing prevents later geometric updates from modifying the classification input.

RelCal: node-preserving relation calibration. After graph compilation, RelCal filters existing edges using their scores and calibrated relation-type thresholds, leaving all nodes and their attributes unchanged.

BoxFix: topology-preserving localization correction. BoxFix applies quality-gated, bounded residuals to normalized center-size boxes in logit space, preserving node identities, types, ordering, and the existing edge list.

DupClean: endpoint-safe node cleanup. DupClean removes eligible low-confidence isolated predictions without deleting endpoints of retained relations. Surviving node attributes and the edge list remain unchanged.

Together, these mechanisms constitute *Guarded Readout*: they revise retained predictions within explicit boundaries, while protected completion expands node and relation support. Detailed state updates, thresholds, and eligibility rules are given in Appendix G.

3.3 Protected completion

Protected completion expands a reference graph $G_0 = (V_0, E_0)$ through three operations: generating complementary proposals, selecting additions around protected objects, and attaching relations on the selected endpoints. The preservation requirements are shared; the proposal and scoring mechanisms are instantiated for each domain.

NodePropose generates a candidate-node bank $\mathcal{B}_{\text{cand}}$ without changing G_0, combining domain-specific visual predictions with pixel/layout or morphology proposals. These explicit priors are system inputs; proposal geometry is not attributed solely to frozen visual representations.

AnchorSelect appends candidates under domain-specific budgets while preserving G_0 and accepted anchors.

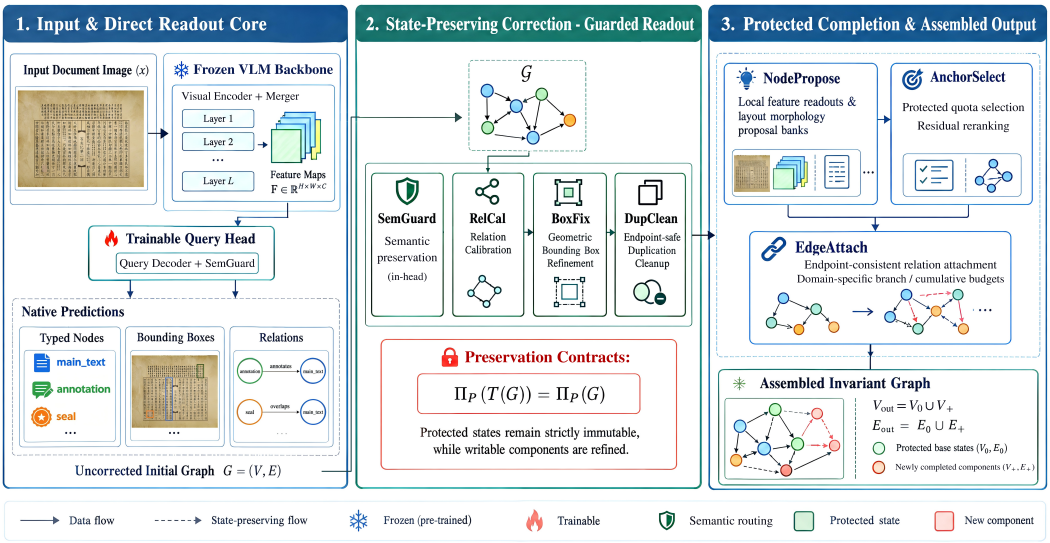


Figure 1: **Overview of Invariant Graph Readout.** Direct readout predicts typed nodes, boxes, and relations from frozen visual representations. SemGuard acts inside the query head; graph-level correction and protected completion follow explicit preservation contracts. Domain configurations and panel interpretation appear in Table 2 and Appendix G.

EdgeAttach scores and inherits relations on the selected node set, requiring every added edge to involve at least one added endpoint. It preserves selected node attributes and protected reference edges.

Candidate sources, selection rules, and budgets are specified in Appendix C.

Let $a_k \in \{0, 1\}$ indicate whether candidate node v_k is accepted. Completion returns

$$\begin{aligned} V_+ &= \{v_k \in \mathcal{B}_{\text{cand}} : a_k = 1\}, \\ V_{\text{out}} &= V_0 \cup V_+, \qquad E_{\text{out}} = E_0 \cup E_+, \\ U_0 &= U(V_0), \quad U_+ = U(V_+), \quad U_0 \cap U_+ = \varnothing, \\ U_{\text{out}} &= U_0 \cup U_+, \\ E_+ &\subseteq (U_{\text{out}} \times \mathcal{R} \times U_{\text{out}}) \setminus (U_0 \times \mathcal{R} \times U_0). \end{aligned} \tag{3}$$

Thus, completion preserves the reference graph while allowing edges that involve added nodes. Inheritance obeys the same endpoint requirement by transferring an incident relation to an added node; it is not an exception permitting new edges solely between original nodes. These constraints govern admissible edits, not the correctness of every added object.

We use *Full* for the complete, domain-configured Invariant Graph Readout system, and *contract-guided completion* for protected completion under these requirements. OracleGraph starts from Guarded Readout; GraphDoc starts from a target-trained global-and-local graph readout baseline, denoted *Base+Local Readout*.

3.4 Algorithm Synthesis

Algorithm 1 summarizes the shared construction rules; domain-specific execution and checks are detailed in Appendix G.

3.5 Training and Selection

We freeze the pretrained visual encoder and merger and fit domain-specific heads and completion learners for three independent pipelines per domain. Reused upstream lineages remain fixed during downstream fitting. OracleGraph uses source-grouped nested predictions, keeping pages and their tiles in the same partition. Training resources and update budgets are detailed in Appendix B.

Table 2: Domain instantiations of the shared graph-edit interface. Completion retains common preservation requirements while starting-graph procedures, proposal sources, trained models, and calibration remain domain-specific.

Shared interface	OracleGraph instantiation	GraphDoc instantiation
Starting graph	Guarded Readout	Target-trained Base+Local Readout with calibrated relations
NodePropose	Local strips and pixel/layout candidates with explicit grid priors	Spatial/local and morphology candidates for boundary and common nodes
AnchorSelect	Conservative anchors and learned residual quota allocation	Text/List-item anchors and matched-count residual utility reranking
EdgeAttach	Domain-specific relation constraints; cumulative edge budget	Native relation ontology; reciprocal units and separate branch budgets

Algorithm 1 Invariant Graph Readout

Require: Image $\boldsymbol{x}$; frozen visual encoder and merger; fitted models; domain configuration d
Ensure: Document graph G_{out}
1: Run query heads on frozen visual features, applying SemGuard where configured before graph compilation.
2: Construct $G_0 = (V_0, E_0)$ using the domain-specific starting procedure in Table 2.
3: Set $G_{\text{out}} \leftarrow G_0$; initialize protected anchors and budgets.
4: **for** each configured completion stage **do**
5: Generate or retrieve candidate banks and scores with NodePropose.
6: Select additions with AnchorSelect, retaining protected nodes and accepted anchors.
7: Assemble selected nodes with unique stable identities.
8: Execute scheduled inheritance and EdgeAttach from the stage's protected reference edges, retaining E_0 and enforcing endpoint, ontology, and budget constraints.
9: Update G_{out} to the stage graph and verify preservation and graph validity.
10: **end for**
11: **return** G_{out}

Fitting and operating-point selection use training and calibration/development resources, with test partitions held out. OracleGraph's primary policy is fixed before confirmation-page generation; two lower-threshold policies are frozen before testing and do not replace it after observing JSON results. GraphDoc uses finite-candidate calibration with structural fallback, fixed before evaluation. Fallback does not imply that the original performance conditions were satisfied. Thresholds and selection rules appear in Appendix C.

4 EXPERIMENTS

Datasets. OracleGraph (Sun et al., 2026) contains historical-style document pages. Our held-out evaluation uses 600 pages, seven content-node categories, and three directed relation types. For GraphDoc (Chen et al., 2025), we use the native eleven node categories and eight relation types. Our custom document-level split is drawn from its official training-source collection; the held-out evaluation contains 1,000 pages from 366 document-identity components, with at most four pages per component. The task is single-page document-graph recovery, not text transcription or cross-page relation prediction.

Models and comparisons. The main experiments use Qwen3-VL-8B-Instruct. Direct readout consumes 4,096-dimensional features from its frozen visual encoder and merger without language-model contextualization. Strict JSON Generation uses the corresponding VLM with LoRA adaptation and greedy autoregressive decoding. We evaluate seeds 42, 7, and 2024. OracleGraph Full uses a union of 5,758 fitting source IDs across modules, compared with 4,700 for Guarded Readout and Strict JSON Generation; this is a complete-system comparison rather than a supervision-matched interface ablation. GraphDoc compares Full with its target-trained Base+Local Readout and a matched-count source-score selection control.

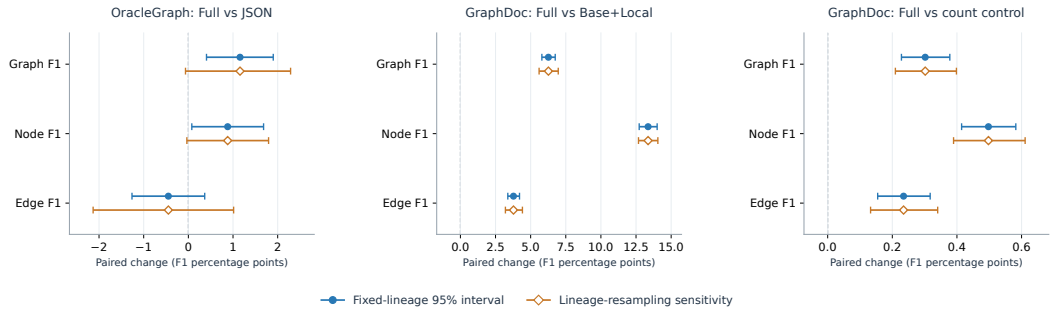


Figure 2: **Paired system differences.** Points denote paired estimates in percentage points. Error bars report 95% bootstrap intervals conditional on the three trained pipelines and lineage-resampling sensitivity intervals, as distinguished in the legend. These are intervals for differences between systems, not separate system-score uncertainties. Horizontal scales differ across panels.

Evaluation. We report dataset-micro Graph, Node, and Edge F1 with type-aware node matching and exact directed relation matches. OracleGraph accepts node matches at $\mathrm{IoU} \geq 0.5$; GraphDoc requires $\mathrm{IoU} > 0.5$. Each main system score averages the dataset-micro scores computed separately for the three trained pipelines. Where reported, $\pm$ denotes their sample standard deviation. Paired uncertainty uses 30,000 bootstrap draws with shared source-page or document-component weights, reporting fixed-lineage intervals and lineage-resampling sensitivity. Appendix A defines the metrics and aggregation.

4.1 Complete-system performance against Strict JSON Generation

We first test the practical graph-construction alternative introduced in Section 1: a complete preservation-aware readout system versus autoregressive structural generation on the same OracleGraph pages (Tables 3 and 4).

Table 3: Results on 600 held-out OracleGraph pages. Scores average dataset-micro results across three training seeds. Graph F1 reports mean $\pm$ sample SD; Node and Edge F1 report means. Paired intervals are given separately in Table 4.

System	Graph F1	Node F1	Edge F1
Strict JSON Generation	0.3872 ± 0.0072	0.538452	0.241427
Guarded Readout	0.3969 ± 0.0020	0.544643	0.236459
Invariant Graph Readout (Full)	0.3988 ± 0.0023	0.547253	0.236952

Table 4: Full minus Strict JSON Generation on OracleGraph. Differences and intervals are in percentage points.

Metric	Difference (pp)	Fixed-lineage 95% CI	Resampling sensitivity
Graph F1	+1.1570	[+0.4062, +1.9008]	[-0.0641, +2.2889]
Node F1	+0.8802	[+0.0773, +1.6842]	[-0.0333, +1.7961]
Edge F1	-0.4475	[-1.2619, +0.3663]	[-2.1321, +1.0137]

Full achieves Graph F1 of 0.3988 ± 0.0023, compared with 0.3872 ± 0.0072 for Strict JSON Generation. The mean paired improvement is 1.1570 percentage points and is positive for each observed seed (Appendix E). Its fixed-lineage 95% interval is $[0.4062, 1.9008]$ points; the lineage-resampling sensitivity interval is $[-0.0641, 2.2889]$. Thus, the observed quality advantage supports the complete-system comparison.

4.2 SUPERVISED REUSE OF CONTRACT-GUIDED COMPLETION ON GRAPHDOC

GraphDoc evaluates supervised reuse of the completion interface with domain-specific starting graphs, ontologies, proposals, and learned components across three independently trained pipelines (Tables 5 and 6). The 1,000 evaluation pages are held out from model fitting; the reported evaluation uses the revised frozen calibration protocol (Appendix D).

Table 5: System performance on GraphDoc.

System (same 1,000 pages)	Graph F1	Node F1	Edge F1
Base+Local Readout	0.044252	0.116378	0.018162
Matched-count source-score control	0.103872	0.244982	0.053587
Invariant Graph Readout (Full)	0.106891	0.249956	0.055936

Table 6: Graph-F1 comparisons on GraphDoc.

Comparison	Diff (pp)	Fixed-lineage 95% CI	Resampling sensitivity
Full minus Base+Local Readout	+6.2639	[+5.7985, +6.7493]	[+5.6012, +6.9637]
Full minus matched-count control	+0.3018	[+0.2283, +0.3780]	[+0.2095, +0.3985]

Full increases mean Graph F1 from 0.0443 for Base+Local Readout to 0.1069, a gain of 6.2639 percentage points, positive across all three pipelines. Node and Edge F1 increase by 13.3578 and 3.7774 points. These results demonstrate supervised reuse of the contract-guided completion interface with domain-specific components rather than transfer of an unchanged OracleGraph pipeline.

Relative to the matched-count source-score control, Full gains a further 0.3018 Graph F1 points. This comparison shares the candidate pool, accepted anchors, realized per-page node counts, and relation pipeline, identifying the conditional effect of residual utility reranking.

Thin-boundary additions and residual selection activate on approximately 96.7% and 68.5% of page-pipeline cases. All 243,000 structural-contract checks pass, providing execution evidence for the declared preservation requirements. The stage trajectory appears in Appendix F. Final Edge precision is 0.08814, and node-count MAE rises from 7.8080 to 8.6723, leaving relation precision and count control as practical limitations.

4.3 CONDITIONAL EFFECTS OF PROTECTED COMPLETION

On the same 600 OracleGraph pages, completion improves Graph F1 over Guarded Readout by 0.1888 percentage points, with a fixed-lineage 95% interval of $[0.1359, 0.2455]$. Table 7 evaluates which selection and attachment choices contribute within this preservation-aware interface. Each comparison changes a specified factor while retaining the applicable protection requirements.

Table 7: Conditional component comparisons on 600 OracleGraph pages and three seeds under the primary policy. Differences are Full minus control, in percentage points; intervals condition on the trained pipelines.

Control	Changed factor	Diff (pp)	95% CI (pp)
Heuristic Completion	Replace learned completion recipe	$+0.6915$	$[+0.6272, +0.7571]$
Source-Score Ranking	Raw source scores at matched source/type counts	$+0.0819$	$[+0.0321, +0.1329]$
Global Verifier Ranking	Global ranking at matched residual count	$+0.0293$	$[+0.0090, +0.0510]$
Fixed-Count Allocation	Fixed residual total with predicted source proportions	$+0.0913$	$[+0.0577, +0.1248]$
Node-Only Completion	Remove all new edges; retain final nodes	-0.0159	$[-0.0365, +0.0054]$
Inheritance-Only	Replay inheritance without learned attachment	-0.0018	$[-0.0107, +0.0077]$

The primary high-threshold policy activates residual allocation. Diagnostic-policy results and the exact count-preservation rules of each control are detailed in Appendix C.1.

Node-matched controls isolate relation-stage effects from changes in node selection. Full differs from Node-Only and Inheritance-Only completion by only -0.0159 and -0.0018 Graph F1 points, respectively, with both intervals spanning zero. These comparisons evaluate attachment within the coupled relation pipeline, where feasibility and budget constraints jointly determine its net effect, rather than assuming an independent additive contribution. Crucially, preservation is a structural requirement, not an accuracy-based assertion: direct graph-state checks verify compliance with the declared invariants in the evaluated executions, while the controlled comparisons quantify the conditional performance effects of completion choices.

4.4 Supporting interface and adaptation evidence

On a separate 600-page OracleGraph set, guarded readout satisfied the preregistered Graph-F1 non-inferiority criterion against strict JSON generation. Additional probes demonstrate viable visual-stage readout on Qwen2.5-VL and Idefics3 without LM contextualization. On M5HisDoc (Shi et al., 2023), separate supervised comparisons yield Graph-F1 gains of 2.858 points for PairAdapt and 0.488 points for target-trained BoxFix, each against its own baseline. The latter gain is achieved without changing node identities or graph topology. These results provide supporting evidence for cross-backbone readout and topology-preserving correction, rather than complete-system transfer (Appendix D).

4.5 End-to-end cost and corresponding quality

We measure batch-one end-to-end latency on the same A100-SXM4-40GB GPU with seed 42, using 100 pages per dataset and three repeats. Timing covers image loading through graph serialization without cached features or candidate banks; quality uses the actual timed outputs (Table 8). Warm-up, timing boundaries, decoding, and resource accounting appear in Appendix H.

Table 8: End-to-end cost and timed-output quality on 100 pages per dataset, with batch size one and three repeats. All systems use the same A100-SXM4-40GB GPU.

Dataset / system	**Mean (s/pg)**	**Median**	**p90**	**Graph F1**
OracleGraph / Guarded Readout	0.8354	0.8137	0.9151	0.392171
OracleGraph / Invariant Graph Readout	3.5974	3.5687	3.9185	0.395750
OracleGraph / Strict JSON Generation	80.1422	72.2738	96.3806	0.382393
GraphDoc / Base+Local Readout	0.7329	0.6378	1.2108	0.042481
GraphDoc / Matched-count control	0.8488	0.8364	1.1860	0.105963
GraphDoc / Invariant Graph Readout	0.8773	0.8579	1.2444	0.108323

On the OracleGraph subset, the ratio of measured mean latencies between Strict JSON Generation and Full is 22.3 (80.14 versus 3.60 s per page), while their timed-output Graph F1 scores are 0.382393 and 0.395750. Strict JSON Generation produces approximately 1,157 tokens per page; Full generates no autoregressive structural tokens. These results demonstrate a favorable measured quality–latency trade-off for the complete graph-native system under the reported execution conditions, rather than a fully isolated hardware speedup. GraphDoc Full averages 0.8773 s per page.

5 Conclusion

We introduced Invariant Graph Readout, a non-autoregressive framework that combines direct graph prediction from frozen visual representations of a pretrained vision-language model with state-preserving correction and protected completion. Its central abstraction is an operation-specific preservation contract: each edit declares what it may change and what it must retain, allowing domain-specific components to operate through a shared graph-edit interface. Together, these results support preservation-aware graph editing as a practical alternative to autoregressive structural generation and demonstrate supervised reuse of contract-guided completion across document domains.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish the academic writing and formatting of this manuscript. We have not used generative AI tools for data synthesis or evaluating the structural models. We have thoroughly reviewed all AI-assisted work, ensuring technical correctness, and take full responsibility for the final content of this work, including text, claims, and formatting artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our research adheres strictly to standard ethical guidelines in machine learning. The datasets utilized, such as OracleGraph and GraphDoc, are standard benchmarks for document layout analysis and do not contain personally identifiable information (PII) or sensitive human-subject data. We confirm full compliance with respective dataset licenses and explicitly state that restricted historical-document images used in adaptation tasks will not be redistributed outside their authorized licenses.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we explicitly detail the operational contracts of our graph edits (Section 3.1) and all hyperparameter boundaries. The evaluation metrics (Graph/Node/Edge F1) are meticulously defined. Source code, network architectures, checkpoint loading protocols, and deterministic evaluation scripts will be released as part of our supplementary material. In Section 4.5 and Appendix H, we transparently log the exact computational setups, batch sizes, hardware specifications (A100 GPUs), and memory bounds used for evaluating inference latency.

REFERENCES

Haoyu Cao, Kun Yin, Yunfei Wu, Bing Liu, Zhongpeng Cai, Xiaotian Li, Huang Chen, Xin Li, Yinsong Liu, Deqiang Jiang, Xing Sun, Yunsheng Wu, Qianyu Li, Antai Guo, Yanzhen Liao, Yanqiu Qu, Haodong Lin, Chengxu He, and Shuangyin Liu. Youtu-Parsing: Perception, structuring and recognition via high-parallelism decoding. *arXiv preprint arXiv:2601.20430*, 2026. URL https://arxiv.org/abs/2601.20430.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pp. 213–229, 2020.

Yufan Chen, Jiaming Zhang, Kunyu Peng, Junwei Zheng, Ruiping Liu, Philip Torr, and Rainer Stiefelhagen. RoDLA: Benchmarking the robustness of document layout analysis models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Chen_RoDLA_Benchmarking_the_Robustness_of_Document_Layout_Analysis_Models_CVPR_2024_paper.html.

Yufan Chen, Ruiping Liu, Junwei Zheng, Di Wen, Kunyu Peng, Jiaming Zhang, and Rainer Stiefelhagen. Graph-based document structure analysis. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=Fu0aggezN9.

Yufan Chen, Omar Moured, Ruiping Liu, Junwei Zheng, Kunyu Peng, Jiaming Zhang, and Rainer Stiefelhagen. HybriDLA: Hybrid generation for document layout analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 3147–3155, 2026. doi: 10.1609/aaai.v40i4.37308.

Hao Feng, Qi Liu, Hao Liu, Jingqun Tang, Wengang Zhou, Houqiang Li, and Can Huang. DocPedia: Unleashing the power of large multimodal model in the frequency domain for versatile document understanding. *Science China Information Sciences*, 67(12):220106, 2024. doi: 10.1007/s11432-024-4250-y.

Xian Guo, Wenjing Ding, Yichuan Wang, and Jie Jiang. SSPRCD: Scene graph-based street-scene spatial positional relation change detection with graph differencing and structural quantification. *ISPRS International Journal of Geo-Information*, 15(4):161, 2026. doi: 10.3390/ijgi15040161.

Anwen Hu, Haiyang Xu, Liang Zhang, Jiabo Ye, Ming Yan, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. mPLUG-DocOwl2: High-resolution compressing for OCR-free multi-page document understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL https://aclanthology.org/2025.acl-long.291/.

Xin Hu, Ke Qin, Wen Yin, Yuan-Fang Li, Ming Li, and Tao He. Can we build scene graphs, not classify them? FlowSG: Progressive image-conditioned scene graph generation with flow matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/papers/Hu_Can_We_Build_Scene_Graphs_Not_Classify_Them_FlowSG_Progressive_CVPR_2026_paper.pdf.

Chen-Yu Lee, Chun-Liang Li, Timothy Dozat, Vincent Perot, Guolong Su, Nan Hua, Joshua Ainslie, Renshen Wang, Yasuhisa Fujii, and Tomas Pfister. FormNet: Structural encoding beyond sequential modeling in form document information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022. URL https://aclanthology.org/2022.acl-long.260/.

Chen-Yu Lee, Chun-Liang Li, Hao Zhang, Timothy Dozat, Vincent Perot, Guolong Su, Xiang Zhang, Kihyuk Sohn, Nikolay Glushnev, Glushnev, Renshen Wang, Joshua Ainslie, Shangbang Long, Siyang Qin, Yasuhisa Fujii, Nan Hua, and Tomas Pfister. FormNetV2: Multimodal graph contrastive learning for form document information extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL https://aclanthology.org/2023.acl-long.501/.

Lei Li, Ze Zhao, Meng Li, Zhongwang Lun, Yi Yuan, Xingjing Lu, Zheng Wei, Jiang Bian, and Zang Li. Efficient document parsing via parallel token prediction. In *Findings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR Findings)*, 2026. URL https://arxiv.org/abs/2603.15206.

Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokula, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. LaViDa: A large diffusion language model for multimodal understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://neurips.cc/virtual/2025/poster/119782.

Jiefeng Ma, Jun Du, Pengfei Hu, Zhenrong Zhang, Jianshu Zhang, Huihui Zhu, and Cong Liu. HRDoc: Dataset and baseline method toward hierarchical reconstruction of document structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 1870–1877, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/25277.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24838–24848, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Ouyang_OmniDocBench_Benchmarking_Diverse_PDF_Document_Parsing_with_Comprehensive_Annotations_CVPR_2025_paper.html.

Paolo Pellizzoni, Till Hendrik Schulz, and Karsten Borgwardt. Gelato: Graph edit distance via autoregressive neural combinatorial optimization. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=6ZTcLNmguc.

Yongxin Shi, Chongyu Liu, Dezhi Peng, Cheng Jian, Jiarong Huang, and Lianwen Jin. M5HisDoc: A large-scale multi-style Chinese historical document analysis benchmark. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 36, 2023.

Joongmin Shin, Chanjun Park, Jeongbae Park, Jaehyung Seo, and Heuiseok Lim. MultiDocFusion: Hierarchical and multimodal chunking pipeline for enhanced RAG on long industrial documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025. URL https://aclanthology.org/2025.emnlp-main.1062/.

Zhihong Sun, Qi Fan, Pan Liu, Yang Yi, and Chen Ye. Benchmarking and optimizing multimodal structured generation: The OracleGraph dataset and PRISM framework. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. URL https://openreview.net/forum?id=mMpgycVI2H.

Pavan Kumar Anasosalu Vasu, Fartash Faghri, Chun-Liang Li, Cem Koc, Nate True, Albert Antony, Gokul Santhanam, James Gabriel, Peter Grasch, Oncel Tuzel, and Hadi Pouransari. FastVLM: Efficient vision encoding for vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. URL https://cvpr.thecvf.com/virtual/2025/poster/32887.

Dongsheng Wang, Natraj Raman, Mathieu Sibue, Zhiqiang Ma, Petr Babkin, Simerjot Kaur, Yulong Pei, Armineh Nourbakhsh, and Xiaomo Liu. DocLLM: A layout-aware generative language model for multimodal document understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024a. URL https://aclanthology.org/2024.acl-long.463/.

Jiawei Wang, Kai Hu, Zhuoyao Zhong, Lei Sun, and Qiang Huo. Detect-order-construct: A tree construction based approach for hierarchical document structure analysis. *Pattern Recognition*, 156:110836, 2024b. doi: 10.1016/j.patcog.2024.110836.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024c. URL https://arxiv.org/abs/2409.12191.

Yuqing Wang, Shuhuai Ren, Zhijie Lin, Yujin Han, Haoyuan Guo, Zhenheng Yang, Difan Zou, Jiashi Feng, and Xihui Liu. Parallelized autoregressive visual generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12955–12965, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wang_Parallelized_Autoregressive_Visual_Generation_CVPR_2025_paper.html.

Zining Wang, Tongkun Guan, Pei Fu, Chen Duan, Qianyi Jiang, Zhentao Guo, Shan Guo, Junfeng Luo, Wei Shen, and Xiaokang Yang. Marten: Visual question answering with mask generation for multi-modal document understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Wang_Marten_Visual_Question_Answering_with_Mask_Generation_for_Multi-modal_Document_CVPR_2025_paper.pdf.

Dongil Yang, Minjin Kim, Sunghwan Kim, Beong-woo Kwak, Minjun Park, Jinseok Hong, Woontack Woo, and Jinyoung Yeo. LLM meets scene graph: Can large language models understand and generate scene graphs? a benchmark and empirical study. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 21335–21360, 2025. URL https://aclanthology.org/2025.acl-long.1036/.

Sanggeon Yun, Raheeb Hassan, Ryozo Masukawa, Nathaniel D. Bastian, and Mohsen Imani. MissionHD: Hyperdimensional refinement of distribution-deficient reasoning graphs for video anomaly detection. *arXiv preprint arXiv:2508.14746*, 2025. URL https://arxiv.org/abs/2508.14746.

A Dataset Ontologies and Evaluation Protocols

This appendix specifies the ontologies, matching rules, and metric aggregation used to evaluate predicted document graphs.

OracleGraph ontology. For OracleGraph (Sun et al., 2026), we predict seven content-node categories: `MAIN_TEXT`, `INTERLINEAR_ANNOTATION`, `TOP_MARGINALIA`, `BOTTOM_MARGINALIA`, `SIDE_MARGINALIA`, `CENTER_FOLD_TEXT`, and `RED_SEAL_STAMP`. The directed relation types are `READS_AFTER`, `ANNOTATES`, and `OVERLAPS`. Structural skeleton objects, such as page borders and column dividers, are excluded from evaluation. Node matching is type-wise Hungarian matching, accepting matches at $\mathrm{IoU} \geq 0.5$.

GraphDoc ontology. For GraphDoc (Chen et al., 2025), we predict eleven native categories: `Caption`, `Footnote`, `Formula`, `List-item`, `Page-footer`, `Page-header`, `Picture`, `Section-header`, `Table`, `Text`, and `Title`. The relation types are `up`, `down`, `left`, `right`, `parent`, `child`, `reference`, and `sequence`. We retain the native instance-matching rule with $\mathrm{IoU} > 0.5$, resolving conflicting matches in favor of the highest IoU.

Full-output scoring. A predicted relation is a true positive only when both endpoints are correctly matched and its directed relation type agrees with the reference graph. All other predicted relations, including those with unmatched endpoints, count as false positives. GraphDoc predictions are deduplicated as directed typed triples before edge scoring. We report F1 at fixed operating points, distinct from GraphDoc's official average-precision metric mAP_g.

For each pipeline, true-positive, false-positive, and false-negative counts are pooled across evaluation pages before computing

$$
\begin{aligned}
F_1^N &= \frac{2\mathrm{TP}_N}{2\mathrm{TP}_N + \mathrm{FP}_N + \mathrm{FN}_N}, \\
F_1^E &= \frac{2\mathrm{TP}_E}{2\mathrm{TP}_E + \mathrm{FP}_E + \mathrm{FN}_E}, \\
F_1^G &= \frac{2(\mathrm{TP}_N + \mathrm{TP}_E)}{2(\mathrm{TP}_N + \mathrm{TP}_E) + \mathrm{FP}_N + \mathrm{FP}_E + \mathrm{FN}_N + \mathrm{FN}_E}.
\end{aligned}
\tag{4}
$$

Graph F1 pools node and edge counts in a single micro-averaged score.

Pipeline aggregation and uncertainty. Let g_s be the dataset-micro Graph F1 of pipeline s. The main three-pipeline comparisons report

$$
\bar{g} = \frac{1}{S}\sum_{s=1}^{S} g_s, \qquad s_g = \sqrt{\frac{1}{S-1}\sum_{s=1}^{S}(g_s - \bar{g})^2}, \qquad S = 3. \tag{5}
$$

The notation $\bar{g} \pm s_g$ reports the mean and sample standard deviation across trained pipelines. For two systems A and B, the paired effect is

$$
\widehat{\Delta} = \frac{1}{S}\sum_{s=1}^{S}(g_s^{\mathrm{A}} - g_s^{\mathrm{B}}). \tag{6}
$$

Each of 30,000 bootstrap draws applies shared source-page or document-component resampling weights across systems and pipelines, recomputing dataset-micro metrics from resampled counts. Fixed-lineage intervals condition on the three trained pipelines; lineage-resampling sensitivity additionally resamples pipeline indices with replacement. Resampling preserves the shared page identity across pipelines.

Resampling units and interpretation. OracleGraph resamples the same 600 source-page identities jointly across systems and seeds. GraphDoc resamples 366 conservative document-identity components, applying each component's weight to all its pages and sharing those weights across systems and lineages. Its evaluation set contains 1,000 pages, with at most four per component. Metrics are recomputed from weighted counts within each lineage before averaging lineage differences. The 2.5th and 97.5th percentiles of the 30,000 paired draws define the reported 95% intervals. Additional lineage resampling uses the same sampled lineage indices on both sides of each comparison. With only three trained lineages, training-population uncertainty remains imprecisely estimated; lineage resampling provides a sensitivity analysis. Percentage-point differences equal 100 times raw F1 differences.

For OracleGraph, the fixed-lineage Graph-F1 interval for Full minus JSON is positive, whereas its lineage-resampling interval includes zero; the Edge-F1 point difference is negative. GraphDoc's evaluation pages are held out from model fitting but had been exposed during earlier design work; Appendix D distinguishes that history from the revised frozen evaluation.

B Detailed Training Configurations and Optimization

The main OracleGraph and GraphDoc experiments use Qwen3-VL-8B-Instruct. Direct readout freezes its visual encoder and merger and consumes 4,096-dimensional final visual features without language-model contextualization. Visual encoding uses bfloat16; downstream tree learners are fitted separately. The additional backbones in Appendix D.2 evaluate the readout interface separately; both reported Full systems use Qwen3-VL-8B-Instruct.

OracleGraph global head. The global query head uses 96 anchored queries on an 8×12 reference grid, four decoder layers, hidden width 512, and eight attention heads. Initial normalized box width and height are 0.05 and 0.15. It is trained on 4,700 source pages for 4,700 single-page updates using AdamW, peak learning rate 2×10^{-4}, weight decay 10^{-4}, 100 warm-up steps, cosine decay, and gradient clipping at 1.0. Classification, L1-box, GIoU, relation, and cardinality loss weights are $1, 5, 2, 5, 2$. The relation focal exponent is 2, the auxiliary-box loss weight is 0.5, and the relation objective includes 64 hard negative pairs. SemGuard retains the pre-refinement decoder state for classification; the two geometric state-update MLPs have zero-initialized final layers. Appendix G specifies the state routing and graph-edit constraints.

BoxFix. The corrector is fitted separately for 4,700 updates with the upstream query head and visual encoder frozen. L1-box, GIoU, and quality loss weights are $5, 2, 1$; residual-identity and degradation penalties have weights 0.05 and 2. Matching is determined from frozen base predictions. The maximum logit residual amplitude is 0.5; the quality-dependent gate and the protected attributes are given in Appendix G.

OracleGraph local proposal head. The local head is initialized from its corresponding global head and retains that architecture. It uses 6,000 single-tile updates from 1,000 source pages, learning rate 10^{-4}, and 100 warm-up steps. Inference processes six vertical strips, each with a half-strip context margin on both sides, clipped to the page. Predictions are projected back to full-page coordinates, and only nodes centered inside the strip core are eligible for addition. Context predictions may still provide relation evidence.

OracleGraph completion resources. The verifier, relation scorer, and quota allocator use 2,000 sources, comprising 1,000 standard and 1,000 dense pages, split into 1,610 fitting and 390 calibration sources. All tiles and candidate/event rows derived from a source stay in the same partition. Each of the three upstream lineages is reused and held fixed during downstream fitting. The nested fitting procedure is detailed in Appendix C.1; its fold exclusions apply to the downstream learners, while reused upstream heads may have seen some of these sources. The union of Full's fitting source IDs across modules is 5,758, compared with 4,700 for Guarded Readout and Strict JSON Generation. The main comparison evaluates complete systems with these different supervision budgets.

GraphDoc instantiation. Each seed, 42, 7, or 2024, trains its own 64-query global head on 663 pages for 5,000 updates. Base-quality calibration and the local stage use 534 fitting and 129 calibration pages from that resource. The local head has 20 queries, hidden width 256, two decoder layers, and eight attention heads. It processes six horizontal bands with overlap 0.33, using 1,068 updates, learning rate 2×10^{-4}, and 50 warm-up steps. Its nine local categories exclude Picture and Table. Geometry-only local control and visual-local heads are fitted under paired initialization and page schedules; the deployed lineage uses its frozen selected local configuration. Later completion models use 3,000 fitting and 600 calibration pages and are fitted independently for each lineage. The revised downstream procedure holds these independently trained upstream assets fixed throughout downstream fitting and evaluation.

Strict JSON Generation. The baseline uses the corresponding pretrained VLM with LoRA adapters of rank 16, alpha 32, and dropout 0, targeting `q_proj`, `k_proj`, `v_proj`, `o_proj`, and the

MLP gate/up/down projections. It uses 4,700 single-page updates, learning rate 2×10^{-5}, weight decay 0, 50 warm-up steps, cosine decay, and gradient clipping at 1.0. Inference is greedy, with at most 3,072 new tokens, repetition penalty 1.05, no-repeat n-gram size 20, and key–value caching. Grammar-constrained decoding and JSON repair are disabled. The primary quality evaluation uses batch size eight for readout and initially four for JSON, recursively halved on out-of-memory errors while retaining every page. The runtime comparison in Appendix H uses batch one for all systems.

Prompt and strict parser. The JSON path uses a structure-only Chinese system instruction and a user message containing the image followed by the schema. Their meaning is: output one JSON object containing node types, bounding boxes, and directed relations, with no transcription or extra text. Node abbreviations are MT, IA, TM, BM, SM, CF, and RS; edge abbreviations are R, A, and O. The requested structure is `{"n":[{"id", "t", "box":[x1,y1,x2,y2]}], "e":[{"s", "d", "t"}]}`. The official chat template supplies the assistant-generation prefix. Strict parsing applies JSON decoding to the entire returned string, requires an object root and list-valued `n`/`e`, and performs no substring extraction or repair. A parse failure becomes an empty predicted graph and remains in scoring. Parsing checks output syntax; graph matching and endpoint scoring assess structural accuracy.

Image processing and export. Both domains use the image processor associated with their frozen visual checkpoint; local crops are projected into the original page frame after encoding. OracleGraph's online local path writes RGB JPEG crops at quality 95 with chroma subsampling disabled. GraphDoc's local path uses the frozen band processor and checks a 7×32 valid visual-token grid per band. Processor/tokenizer configuration identity was checked before the added JSON comparison. These processing configurations are domain-specific.

Graph compilation. The OracleGraph base compiler uses node and relation thresholds of 0.5, same-type NMS with IoU threshold 0.5, at most 40 nodes, and at most 16 outgoing edges per source before the configured ontology constraints. RelCal subsequently uses 0.575. The compiler can retain fewer nodes than the 96 predicted queries, and completion can add nodes after compilation. GraphDoc's initial global export uses a 64-node limit and node/relation thresholds of 0.5, followed by the calibrated Base+Local operating points in Appendix C.

Task-specific model size. OracleGraph Guarded Readout has 24.78M neural task parameters; Full has 48.50M plus deployed tree models. Strict JSON Generation has 43.65M LoRA parameters. GraphDoc's global and local heads together contain 24.33M parameters, in addition to downstream tree models and calibrators. These counts cover task-specific parameters and exclude the frozen backbone.

C Completion Learners and Calibration Grids

Completion combines local proposal heads with domain-specific candidate verification, quota allocation, residual ranking, and relation scoring. The pretrained visual backbone remains frozen. The shared interface constrains what can change; candidate families, fitted models, and numerical operating points remain domain-specific.

C.1 OracleGraph candidates and nested fitting

Candidate banks and targets. NodePropose combines local-strip predictions with pixel/layout proposals obtained from thresholding, rule removal, connected components, and geometric zones. Pixel-filtered character-grid and center-fold proposals derive geometry from explicit OracleGraph layout priors. A candidate has a stable source key, type, box, and source score. Same-type overlap with an existing node at $\mathrm{IoU} \geq 0.5$ rejects the candidate and preserves the existing node. Candidate-bank deduplication at $\mathrm{IoU} \geq 0.8$ prioritizes local-strip proposals, with deterministic score/key ordering. The learned completion evaluation covers MT/TM, the types with residual positive supervision.

The verifier, new-endpoint relation scorer, and quota allocator use 45-, 57-, and 33-dimensional features, respectively. Verifier inputs include source/type indicators, source scores, normalized geometry, pixel measurements, overlap/proximity, local degrees, and page counts. Its training label is

a typed one-to-one match to a gold node unmatched by the fixed starting graph. Inference uses the listed prediction, image, and geometry features only. Relation features additionally encode endpoint geometry, original graph degrees, verifier scores, and local relation evidence. Relation events touch at least one candidate endpoint. Sampling and balanced class weights define discriminative scores; their posterior-probability calibration remains unverified.

Protected prefix and allocation. The conservative prefix accepts at most eight TM and two MT nodes, using the selected verifier threshold and same-type conflict rejection. Residual allocation uses three source/type buckets: local-strip MT, local-strip TM, and morphology TM. Its 35 possible classes are

$$\mathcal{Q} = \{(q_1, q_2, q_3) \in \mathbb{Z}^3_{\geq 0} : q_1 + q_2 + q_3 \leq 4\}. \tag{7}$$

The allocator summarizes the starting graph, protected prefix, source/type counts, verifier-score distributions, relation-score distribution, and anchor counts. Candidates are ordered within each bucket by verifier score, source score, and stable key. Conflict rejection can reduce realized counts below requested quotas, and residual additions cannot replace the protected prefix. On fitting pages, all 35 allocations are evaluated after attachment. The target maximizes strict Graph F1, followed by Edge F1, Node F1, fewer realized extra nodes, smaller requested budget, and the fixed lexicographic allocation tie-break. This per-page training target is distinct from the dataset-micro objective used for deployment-policy selection.

Nested fitting. Within the 1,610 fitting sources, three source-grouped outer folds supply held-out predictions. Each outer relation model receives verifier predictions from inner models excluding both the inner held fold and that relation model's outer held fold. There are thirteen verifier fits per lineage: one final model, three outer models, and nine inner models. Three outer relation models provide fitting-page scores; the final relation model uses the final verifier on calibration pages. The allocator is fitted directly to the nested held-page utility targets. Only final-fit models are deployed. Source pages and their tiles never cross a downstream fitting/calibration or fold boundary, although reused upstream heads may have seen some of these sources.

The three learners use histogram gradient boosting with 120 iterations, learning rate 0.05, at most 15 leaves, L2 regularization 5, balanced class weights, and row-random early stopping disabled. Minimum leaf size is 80 for verification and relation scoring and 20 for allocation. All positives and deterministic stratified negative samples are retained for verifier/relation fitting.

Frozen policies and attachment. The finite verifier/relation threshold pairs are $(0.8, 0.95)$, $(0.9, 0.975)$, and $(0.95, 0.99)$. Selection uses the three-lineage mean of standard-development dataset-micro Graph/Edge/Node F1, fewer added objects, and policy identifier; the dense subset does not select the deployment point. The primary policy is $(0.95, 0.99)$. Both lower-threshold policies were separately fixed before the completion confirmation evaluation to exercise conservative stages; they do not replace the primary policy after test results are observed. Completion confirmation and the later added comparison against JSON have different chronological roles (Appendix D).

For fixed final nodes, attachment restarts from the protected starting edges. It processes scored events involving added TM nodes (branch cap eight), geometric inheritance (branch cap six), and remaining scored events, all within a cumulative cap of twelve new directed edges. Inheritance transfers an incident relation from a same-type original node whose box contains the new center and covers at least half of the new box. Typed endpoint, degree, and reading-order constraints remain active. OracleGraph accepts individual directed edges under its own ontology constraints. Final replay can change which inherited events remain feasible under the shared budget, coupling the effects of learned attachment and inheritance.

Conditional completion controls. The primary high-threshold policy activates residual allocation. Two predefined lower-threshold policies additionally exercise conservative completion stages, with Graph F1 improvements of 0.1238 and 0.1621 points over Guarded Readout. Source-Score Ranking preserves anchors and realized source/type counts; Global Verifier Ranking preserves the residual count but can change the source/type mixture. Fixed-Count Allocation preserves predicted source proportions and zero-allocation pages.

C.2 GRAPHDOC LEARNERS AND FINITE CANDIDATE SETS

Learners. The base-quality calibrator uses 69 features and histogram gradient boosting with at most 80 iterations, learning rate 0.08, 15 leaves, minimum leaf size 80, L2 regularization 2, and balanced fitting-query weights. The base/base and local-involved relation models use at most 300 iterations and 31 leaves, with learning rates $0.06/0.05$, minimum leaf sizes $20/10$, and L2 penalties $2/3$. They use balanced class weights and early stopping with a 0.15 validation fraction and patience $15/20$, respectively. Their calibrated decoder accepts reciprocal structural relations and limits outgoing edges per source to 16; local-involved relation types are up, down, and sequence.

Thin-boundary node classifiers use 80 iterations, learning rate 0.05, seven leaves, minimum leaf size 100, and L2 regularization 5. Thin-node relation scoring uses at most 100 iterations, learning rate 0.05, 15 leaves, minimum leaf size 80, and L2 regularization 5. Text/List-item verification and relation scoring use 169 and 115 features, respectively, with at most 120 iterations, learning rate 0.05, 15 leaves, minimum leaf size 80, and L2 regularization 5. Residual utility ranking uses a 246-feature ExtraTrees classifier with 300 trees, maximum depth 12, minimum leaf size 8, maximum-feature fraction 0.7, and balanced class weights. Its target indicates whether appending a candidate to the protected two-anchor state increases matched node true positives.

Candidate sources and budgets. Spatial/local predictions and morphology-derived proposals supply thin-boundary and common-node candidates. Thin-boundary caps are selected from $\{1, 2\}$ for Page-header and Page-footer and $\{1, 2, 3, 4\}$ for Section-header; all three reported pipelines select $2, 2, 4$. Text/List-item completion retains up to two anchors and at most one residual node. The matched-count control shares the candidate pool, anchors, realized per-page node counts, and relation pipeline, replacing only residual utility ordering with source-score ordering. Thin-boundary attachment, inheritance, and Text/List-item attachment have separate directed-edge caps of eight, six, and twelve. Reciprocal units are accepted atomically within each separate branch budget.

Finite grids. For a finite score vector S, let

$$\mathcal{T}_K(S) = \{\text{quantile}(S, j/(K-1)) : j = 0, \ldots, K-1\}. \tag{8}$$

Repeated threshold values are deduplicated. Quantiles use only calibration scores, under a grid-construction rule fixed in advance. The threshold 1.1 supplies a disabled-addition candidate to the calibration search; fallback requires an active candidate.

Table 9: GraphDoc calibration candidate construction. Thresholds are generated separately for each trained lineage.

Stage	Finite candidate set
Base-quality node	$\mathcal{T}_{101}(S) \cup \{0.5\}$.
Local node	$\mathcal{T}_{201}(S) \cup \{1.1\}$, evaluated in descending threshold order.
Base/base and local relations	Separate scope-specific $\mathcal{T}_{101}(S) \cup \{0.5, 1.1\}$, evaluated in ascending order.
Thin-boundary nodes	Per-type $\mathcal{T}_{151}(S) \cup \{1.1\}$ crossed with the caps above; joint candidates use the retained per-type pools described below.
Thin-node attachment	$\mathcal{T}_{101}(S) \cup \{0.5, 1.1\}$ with the same global eight-edge decoder used at inference.
Text/List-item anchors	Per-type $\mathcal{T}_{101}(S) \cup \{1.1\}$ and retained-pool threshold pairs; deployed anchor cap two. Caps $4, 6, 8, 12$ are diagnostic only.
Text/List-item attachment	$\mathcal{T}_{101}(S) \cup \{0.5, 1.1\}$ with a global twelve-edge cap for this branch.
Residual ranking / inheritance	Fixed ranker with per-page matched count / fixed geometric rule with cap six; no additional performance-gate search.

For each thin-node type, all threshold/cap candidates are scored; the top ten under each of Node F1, no-edge Graph F1, Node precision, Node recall, and lower count MAE are pooled with one disabled candidate and deduplicated. The Cartesian product of the three pools defines joint thin-node candidates. Text/List-item anchors use the same five-ranking top-ten construction separately for each type and cap, followed by threshold-pair enumeration. The search is restricted to these finite candidate pools. Here, no-edge Graph F1 scores the predicted nodes with an empty predicted edge set, so all gold edges remain false negatives.

C.3 ELIGIBILITY, FALLBACK, AND SELECTED OPERATING POINTS

Starting-node calibration. The base-quality threshold maximizes Node F1 subject to count MAE increasing by at most 1.0 over the objectness-0.5 baseline, with ties resolved by higher precision, higher recall, lower count MAE, and fewer predicted nodes; exact remaining ties retain the first ascending threshold. Local-node selection permits count MAE increase at most 1.0 and precision decrease at most 0.01 relative to the no-local graph. It maximizes Node F1, then no-edge Graph F1, lower count MAE, and higher threshold. These upstream procedures stop if no eligible point exists.

Original downstream performance conditions. Table 10 specifies the conditions retained as preferred eligibility criteria under the revised protocol. All differences use calibration data and raw $[0, 1]$ score units. Structural legality and the stage's preservation constraints are required independently of these performance conditions.

Table 10: Preferred GraphDoc downstream calibration conditions. Δ is relative to the stage's fixed baseline; F_1^{G0} denotes no-edge Graph F1, P precision, and R recall.

Stage / baseline	Conditions
Base relations / no-edge graph	$\Delta F_1^G \geq 0.0001$ and $R_E \geq 0.002$.
Local relations / learned base relations	$\Delta F_1^G \geq 0.0001$, $\Delta F_1^E \geq 0$, $\Delta R_E \geq -0.002$, and at least one true-positive local-involved edge.
Thin nodes / Base+Local	$\Delta F_1^N \geq 0.001$, $\Delta P_N \geq -0.01$, $\Delta R_N \geq 0$, $\Delta F_1^{G0} \geq 0.0003$, and $\Delta\mathrm{MAE} \leq 1$.
Thin attachment / thin-node graph	$\Delta F_1^G \geq 0.001$, $\Delta F_1^E \geq 0.001$, $\Delta R_E \geq 0$, $\Delta P_E \geq -0.005$, at least 20 true-positive thin relations, and at most eight additional edges per page.
Text/List anchors / preceding graph	$\Delta F_1^N \geq 0.005$, $\Delta P_N \geq -0.015$, $\Delta R_N > 0$, $\Delta F_1^{G0} \geq 0.003$, $\Delta\mathrm{MAE} \leq 1$, at least 20 new true-positive nodes of each type, and per-page cap two.
Text/List attachment / inherited graph	$\Delta F_1^G \geq 0.0003$, $\Delta F_1^E \geq 0.001$, $\Delta P_E \geq 0$, $\Delta R_E \geq 0$, and at least 15 new true-positive relations.

Among eligible base-relation candidates, the lexicographic objective is Graph F1, Edge F1, Edge recall, Edge precision, fewer edges per page, and higher threshold. The local-relation objective additionally includes local true-positive edge count before the final threshold tie-break. Eligible thin and Text/List attachment candidates prioritize Graph F1, Edge F1, Edge precision, fewer edges, and higher threshold. Eligible node candidates prioritize Node F1, no-edge Graph F1, Node precision, and lower count MAE, retaining the first enumerated candidate on exact ties.

Predefined structural fallback. If no downstream candidate satisfies its preferred performance conditions, the revised protocol selects among structurally legal, active candidates from the same finite set. Relation-stage fallback maximizes Graph F1, then Edge F1, then higher threshold. Node-stage fallback maximizes Node F1, no-edge Graph F1, Node precision, and lower count MAE; remaining ties use descending canonical sorted-key threshold/cap serialization. Objective comparisons are rounded to twelve decimals, while eligibility uses the original unrounded inequalities. An active node candidate adds nodes; an active relation candidate adds the relevant relation events. Missing or changed required assets, non-finite models/outputs, missing necessary fitting classes, leakage, preservation or matched-count violations, or absence of a legal active candidate terminate the affected procedure. Fallback selects a legal active operating point when the original performance conditions remain unsatisfied.

The local-involved relation branch has zero eligible candidates and uses fallback in each reported lineage. Base relations, thin nodes, thin attachment, the selected two-anchor policy, and Text/List attachment select eligible candidates in all three lineages. Residual ranking and inheritance have no separate performance-gate selection. The thin-attachment calibration uses the same global eight-edge reciprocal-atomic decoder as inference; the earlier per-source/global decoding mismatch was corrected before the revised fitting runs. Appendix D retains the earlier stopping branch and the scope of this revision.

Table 11: Selected GraphDoc operating thresholds, rounded to six decimals for display. Inference uses saved full-precision values.

Component	Seed 42	Seed 7	Seed 2024
Base-quality node	0.574519	0.573557	0.563523
Local node	0.153287	0.190505	0.158806
Base relation	0.897411	0.809363	0.668707
Local-involved relation	0.975751	0.637852	0.566038
Page-header	0.773131	0.365091	0.357549
Page-footer	0.085569	0.096199	0.117758
Section-header	0.127413	0.127771	0.105564
Thin-node attachment	0.616156	0.703160	0.656825
Text candidate	0.898323	0.900845	0.905205
List-item candidate	0.951199	0.884887	0.954866
Text/List-item attachment	0.871086	0.739840	0.787618

D Earlier Evidence and Supporting Experiments

D.1 Independent interface evidence and evaluation chronology

The historical OracleGraph interface comparison uses a different 600-page set from Section 4.1 (see Table 12).

Table 12: Historical system performance on OracleGraph.

Historical system	Graph F1	Node F1	Edge F1
Strict JSON Generation	0.38855	0.53921	0.24422
Guarded Readout	0.39430	0.54006	0.23563

Its Graph-F1 difference is +0.00575, with the originally reported 95% interval [-0.00472, +0.01584], satisfying the preregistered -0.010 non-inferiority criterion. The interval includes zero. All 1,800 direct-readout outputs are constructible with zero autoregressively generated tokens. This interval and the historical 89.01 latency ratio apply specifically to Guarded Readout. The current OracleGraph Full-versus-JSON comparison was added after the full-system completion results were known, without retraining or threshold reselection. Its models and primary policy remain fixed; the added baseline comparison is retrospective.

The earlier GraphDoc synthesis evaluates 300 pages with 84 document names under fixed upstream training (summarized in Table 13). It is a follow-up synthesis after the correction confirmation.

Table 13: Historical system performance on GraphDoc synthesis.

Historical GraphDoc stage	Graph F1	Node F1	Edge F1
Base+Local Readout	0.05586	0.14312	0.01727
Add thin-boundary nodes	0.09455	0.24819	0.01727
Add thin-node relations	0.10558	0.24819	0.03715
Fixed-count completion (cap 3)	0.13608	0.28448	0.06978
Invariant Graph Readout	0.13772	0.28565	0.07164

The historical +8.185-point gain, the earlier +8.042-point synthesis on another set, and the new +6.2639-point full-lineage result are reported as three separate comparisons. In a separate 396-page fixed-upstream replication, the reranking Node-F1 interval remains positive under downstream-seed resampling while the Graph-F1 interval crosses zero.

The original complete-lineage protocol later stops at local-involved relation calibration and produces no final complete-system evaluation. That historical stop remains a failed calibration branch.

Section 4.2 reports a separately frozen revision that explicitly permits finite calibration fallback and completes all three lineages. The 1,000 evaluation pages belong to 366 conservative document-identity components and had been exposed in earlier design work. For the revised execution, evaluation gold is isolated until all three lineages and comparison arms have saved their predictions, and operating points are fitted on the designated calibration resources. The revised fitting procedure excludes evaluation gold and uses local-relation fallback in every lineage. Its results retain the prior-exposure scope and the failed original performance gate.

D.2 Frozen-representation and backbone readout

Backbone and representation comparisons use 300 OracleGraph pages and seeds 42, 7, and 2024. Each condition trains its readout on the specified frozen feature source. V denotes visual-stage features; $V \to L$ denotes image-position states after frozen language-model contextualization; $V \to L{+}P$ also includes prompt-tail states. The controlled representation conditions share the graph head, training pages and order, 4,700 updates, decoding thresholds, node limit, and evaluation batch size. These experiments evaluate the direct-readout interface alone.

Table 14: Additional frozen-representation and backbone comparisons. Scores are means across the three trained readouts.

Setting	Graph F1	Node F1	Edge F1
Qwen3-VL V	0.34942	0.50709	0.18733
Qwen3-VL $V \to L$	0.36842	0.51913	0.21464
Qwen3-VL $V \to L{+}P$	0.36662	0.53567	0.19354
Qwen2.5-VL-7B visual	0.36385	0.52294	0.19915
Idefics3-8B-Llama3 visual	0.35862	0.51382	0.19945

$V \to L$ improves Graph F1 over V by 1.900 percentage points, with a 95% interval of $[0.582, 3.228]$. Adding prompt-tail states changes Graph F1 by -0.180 points relative to $V \to L$, with interval $[-1.510, 0.943]$. Qwen2.5-VL and Idefics3 have higher point scores than Qwen3-VL V, but their difference intervals are $[-1.613, 5.351]$ and $[-5.377, 5.964]$ points. Idefics3 does not satisfy the specified -5.0-point non-inferiority margin. These results establish usable visual-stage readout on multiple frozen stacks. Language contextualization improves the Qwen3-VL readout in this comparison, while cross-backbone equivalence remains unestablished.

For Idefics3, only the SigLIP visual tower and modality connector are run. Its official processor supplies a 3×4 local-crop grid; excluding the duplicate global view gives 2,028 connector tokens per page. The same 23.718M-parameter graph-head configuration is trained for 4,700 steps using the corresponding seed-specific page orders. This supplies cross-family evidence for the readout interface on the evaluated frozen stacks.

Capacity-matched geometric control. A separate lightweight-head probe compares final visual features with spatial-only input under the same pages, 4,700-step budget, decoding policy, and seeds. Graph F1 is 0.34622 for final features and 0.36497 for spatial-only input, a difference of 1.875 points with interval $[0.306, 4.302]$. Spatial-only zeros aligned content features while retaining image-derived grid geometry, two-dimensional position encoding, and anchored queries. An output audit finds one exact graph per seed across its 300 pages. The result reveals a strong geometric prior in this probe configuration; the visual-content contribution in Full remains unisolated by this probe. The capacity-matched final probe is also distinct from V in Table 14.

D.3 M5HisDoc adaptation and endpoint-preserving correction

Dataset and graph construction. M5HisDoc (Shi et al., 2023) is a Chinese historical-document benchmark with 4,000 regular images and 4,000 corresponding hard images generated through rotation, distortion, and resolution reduction. We construct a coarse text-line graph: each annotated line becomes an MT node, its full polygon determines an axis-aligned bounding box, and consecutive lines in the annotated reading order define directed R relations. Source identity and line index align nodes across regular and hard views, with view-specific geometry. Supervision covers text-line lo-

calization and reading order; fine-grained semantic roles and other relation types are outside this evaluation.

Source splits and evaluation units. The adaptation resource starts from the 2,000 official training sources. The 96-query capacity criterion retains 1,797 eligible training-source pairs. A fixed source-ID hash ordering partitions the official validation sources into 400 selection sources and 600 held-out sources. Both views of each source remain in the same partition, and the official test set is excluded from these adaptation and correction experiments. PairAdapt uses 399 usable development sources from the nominal 400-source pool and evaluates seed 2024 on all 600 held-out sources, yielding 1,200 views. The separate BoxFix study uses 400 selection sources and evaluates seeds 2024 and 314159 on the same held-out source pool. Each comparison has its own baseline and seed aggregation.

Paired-view adaptation. PairAdapt initializes a domain-specific MT/R head from the source-trained graph head and keeps the Qwen3-VL visual encoder and merger frozen. It uses 96 queries on an 8×12 anchor grid, combining three deepstack feature levels and the final visual level with four learned memory-level embeddings. Training uses 1,797 paired updates, two views per update, learning rate 10^{-4}, weight decay 10^{-4}, 100 warm-up steps, cosine decay, and gradient clipping at 1.0. Hungarian matching is performed separately for each view; shared target-line identities then align matched queries. From step 100, consistency compares MT probabilities, normalized box residuals relative to each view's own target, and relation probabilities, with weights 0.25, 0.5, and 0.25. Exact-image pairs receive no consistency loss. Node and relation decoding thresholds and the same-type NMS IoU threshold are all 0.5. Development seeds 42 and 7 select the paired, multi-level-memory configuration before the seed-2024 held-out evaluation.

Target-trained BoxFix. The correction study fits a zero-initialized, three-layer, width-512 box head on each fixed PairAdapt base. Inputs are detached query features, base-box geometry, and node probabilities. Training uses the same 1,797 source pairs for 1,797 two-view updates, learning rate 2×10^{-4}, weight decay 10^{-4}, 100 warm-up steps, and gradient clipping at 1.0. L1-box, GIoU, quality, residual-identity, and degradation loss weights are $5, 2, 1, 0.05, 2$; near-miss matches at IoU $[0.3, 0.5)$ have weight 4. Matching is fixed by base predictions. The selected full-residual policy applies

$$\widetilde{\beta}_i = \sigma\big(\sigma^{-1}(\beta_i) + 0.5 \tanh(h_i)\big) . \tag{9}$$

Its residual scale is one. The quality-soft alternative is evaluated only during selection. Inference replaces the coordinates of retained nodes while preserving their identities, types, order, and all edges; the M5 experiment uses BoxFix alone after PairAdapt.

Table 15: Separate M5HisDoc comparisons, each against its own baseline. Baseline and method columns are Graph F1; differences and intervals are in percentage points. PairAdapt uses seed 2024; BoxFix averages seeds 2024 and 314159.

Comparison	Baseline	Method	Diff (pp)	95% CI
PairAdapt versus adapted baseline	0.09533	0.12391	$+2.858$	$[+2.393, +3.353]$
PairAdapt + BoxFix versus its PairAdapt baseline	0.12494	0.12982	$+0.488$	$[+0.323, +0.655]$

Scoring and uncertainty. Strict matching uses IoU 0.5 and directed endpoint/type agreement. Dataset-micro counts pool the two views and all evaluated sources within each seed. Both comparisons use 30,000 source-cluster bootstrap draws and two-sided percentile 95% intervals, retaining both views together and pairing method and baseline within each draw. The historical BoxFix analysis samples source clusters separately within each fixed seed and averages the two seed-level effects per draw. This differs from the shared cross-lineage source weights used in the main experiments. The reported M5 intervals retain that original estimator. BoxFix improves Graph F1 from 0.12391 to 0.12809 for seed 2024 and from 0.12597 to 0.13156 for seed 314159.

Selection chronology and observed scope. PairAdapt was selected before its sealed evaluation. The later BoxFix study found no eligible candidate under the original broad gate, which included

IoU-gap behavior. Before producing any BoxFix held-out output, the study froze a narrower BoxFix-only hypothesis with strict Graph F1 as its primary endpoint and selected the full-residual policy using development seeds 42 and 7. Training, selection, and held-out pools are source-disjoint. Historical models had already scored these held-out sources, so this correction study provides fresh-training, train-disjoint validation on previously evaluated pages. The original broad gate remains failed.

Across 2,400 view–seed evaluations, BoxFix preserves every node identity, node order, and edge, with zero node or edge additions or removals. Its descriptive Graph-F1 changes are $+1.192$ points on regular views and -0.220 on hard views. The broader IoU-gap hypothesis remains unsupported. These experiments demonstrate supervised target-domain readout adaptation and topology-preserving localization correction; the complete protected-completion system is evaluated separately on OracleGraph and GraphDoc.

E LINEAGE-SPECIFIC PRIMARY SCORES

All entries are dataset-micro Graph F1 on the same pages within each dataset (Table 16 and Table 17). Seed labels bind separately trained pipelines, with scores aggregated within each dataset.

Table 16: Lineage-specific scores on OracleGraph.

Seed	JSON	Guarded	Full	Full - JSON (pp)
42	0.380634	0.397009	0.399680	+1.9046
7	0.394913	0.394829	0.396178	+0.1265
2024	0.386089	0.398846	0.400489	+1.4400

Table 17: Lineage-specific scores on GraphDoc.

Seed	Base+Local	Matched-count control	Full	Full - Base (pp)
42	0.042908	0.108069	0.110780	+6.7872
7	0.046485	0.105941	0.109257	+6.2771
2024	0.043362	0.097607	0.100635	+5.7273

The three observed lineage differences are positive. The main tables report both fixed-lineage intervals and lineage-resampling sensitivities to describe conditional and training-related uncertainty.

F GRAPHDOC STAGE TRAJECTORY

This descriptive trajectory follows the full-system branch on the 1,000-page replication set, averaged after computing each lineage's dataset-micro score. The matched-count alternative in Section 4.2 retains the candidate pool, accepted anchors, realized per-page node counts, and final relation pipeline while replacing utility reranking with original source-score residual selection (Table 18). Figure 3 provides the corresponding visualization. The evaluation retains the revised-calibration and prior-exposure scope in Appendix D.

Adding nodes can change endpoint matching, so node count error and intermediate Edge F1 can worsen while the original graph remains unchanged. The final positive system difference is reported together with these costs.

G DETAILED ALGORITHM AND PRESERVATION CHECKS

This appendix expands Algorithm 1 for the domain-specific execution sequences. A configuration specifies the starting-graph procedure, proposal sources, trained learners, protected anchors, ordered

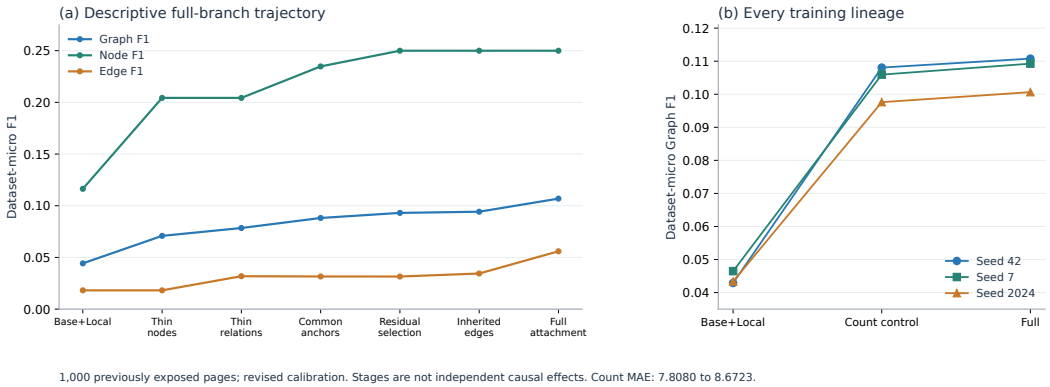


Figure 3: GraphDoc stage trajectory and pipeline-specific results. Left: mean per-lineage dataset-micro Graph, Node, and Edge F1 along the configured Full branch. Right: Base+Local, matched-count control, and Full for each trained lineage. The 1,000 pages were previously exposed during design and evaluated under revised calibration. Stages describe sequential execution with interacting component effects; node-count MAE increases from 7.8080 to 8.6723.

Table 18: Descriptive trajectory of GraphDoc stages.

Full-system stage	**Graph F1**	**Node F1**	**Edge F1**	**Count MAE**
Base+Local Readout	0.044252	0.116378	0.018162	7.8080
Thin-boundary completion	0.070856	0.204315	0.018162	8.0340
Thin-node relation attach	0.078461	0.204315	0.031821	8.0340
Common-node anchor (cap 2)	0.088158	0.234773	0.031582	8.4420
AnchorSelect residual rerank	0.093068	0.249956	0.031530	8.6723
Inheritance on selected ends	0.094222	0.249956	0.034437	8.6723
Full EdgeAttach	0.106891	0.249956	0.055936	8.6723

stages, and budget semantics. The contracts preserve the predicted state designated by each operation. Agreement with ground truth is assessed separately through graph matching.

Framework figure. In Figure 1, SemGuard separates semantic and geometric state inside the trainable query head; its middle-panel box summarizes this in-head mechanism. The correction chain shown corresponds to the OracleGraph starting system. GraphDoc uses a target-trained Base+Local Readout before the shared completion interface. Table 2 lists the domain configurations.

G.1 In-head routing and correction details

SemGuard. Let $Z^{(0)}$ be the trainable query decoder output and $\beta_i^{(0)} \in (0,1)^4$ its initial normalized center-size box. The semantic prediction reads the original state; box-conditioned residual MLPs update the geometric state:

$$\begin{aligned} p_i &= \mathrm{softmax}(\Phi_{\mathrm{sem}}(Z_i^{(0)})), \\ Z_i^{(\ell+1)} &= Z_i^{(\ell)} + \psi_\ell([Z_i^{(\ell)}; \eta_\ell(\beta_i^{(\ell)})]), \\ \beta_i^{(\ell+1)} &= \sigma\Big(\sigma^{-1}(\beta_i^{(\ell)}) + \rho_\ell(Z_i^{(\ell+1)})\Big). \end{aligned} \tag{10}$$

The OracleGraph head uses $\ell = 0, 1$. Here η embeds the current box, ψ updates geometric state, and ρ predicts the box residual. These residual MLP updates follow the attention-based query decoder. Relations read refined state; classification bypasses these subsequent geometric updates. No stop-gradient is applied to $Z^{(0)}$: the shared decoder still receives classification gradients. The invariant concerns forward state routing before graph compilation; shared decoder parameters remain trainable.

RelCal, BoxFix, and DupClean. After compilation, RelCal keeps $E_{\mathrm{cal}} = \{e \in E : s_e \geq \tau_{r(e)}\}$ and leaves every node unchanged. OracleGraph uses $\tau_r = 0.575$ for its three relation types. BoxFix predicts an unconstrained $h_i \in \mathbb{R}^4$ and quality q_i, applying

$$\widetilde{\beta}_i = \sigma\big(\sigma^{-1}(\beta_i) + g(q_i)\, 0.5 \tanh(h_i)\big), \qquad g(q_i) = \sigma\left(\frac{0.55 - q_i}{0.10}\right). \tag{11}$$

The coordinate residual is bounded in box-logit space. The corrected box is converted to image coordinates, while node identity, type, order, and existing relations remain unchanged. DupClean removes only eligible isolated predictions: degree-zero INTERLINEAR_ANNOTATION nodes with confidence below 0.925, processed in ascending confidence order, with at most eight removals per page. If $\mathcal{D}$ is the deleted identity set, $\{u, v\} \cap \mathcal{D} = \varnothing$ for every retained edge (u, r, v). Surviving attributes and edges remain unchanged.

G.2 Configured construction and completion

OracleGraph's conservative stages comprise boundary nodes, boundary attachment, common nodes, inheritance, and learned attachment, followed by residual quota selection and final attachment replay. The primary high-threshold policy can leave conservative stages inactive. Final attachment is replayed from E_0 on final selected nodes, recomputing the added edges under the current constraints. Therefore, relation branches can interact through feasible endpoints, degree limits, and the shared cumulative edge budget.

GraphDoc starts from its calibrated Base+Local graph, then adds thin-boundary nodes and their relations, protects common-node anchors, selects a residual node, and applies inheritance and common-node attachment. Its source-score control and utility-reranked branch protect the same anchors and match realized added-node counts per page; their final relation pipeline is held fixed. The separate branch budgets and reciprocal constraints are given in Appendix C.

G.3 Protected state and execution checks

For completion, $V_{\mathrm{out}} = V_0 \cup V_+$ and $E_{\mathrm{out}} = E_0 \cup E_+$, with disjoint old and new identities. Every added edge must touch a new endpoint. Inherited edges obey the same new-endpoint requirement.

Algorithm 2 Detailed domain-configured Invariant Graph Readout

Require: Image $\boldsymbol{x}$; frozen visual encoder and merger; trained heads and completion models; domain configuration d
Ensure: Predicted document graph G_{out}
Extract the configured visual features and run query heads, using SemGuard inside enabled heads before graph compilation.
if $d = \text{OracleGraph}$ **then**
  Compile predictions and apply RelCal, BoxFix, then DupClean.
else
  Construct the target-trained Base+Local graph with its calibrated nodes and relations.
end if
Denote the resulting protected starting graph by $G_0 = (V_0, E_0)$; initialize the current graph to G_0.
Initialize protected anchors and the configured per-stage or cumulative budgets.
for each configured completion stage s **do**
  Generate or retrieve candidate banks and scores using NodePropose and the fitted models.
  Apply AnchorSelect with quotas or residual ranking, retaining protected nodes and accepted anchors.
  Assemble the selected nodes with unique stable identities distinct from retained identities.
  Execute scheduled inheritance and EdgeAttach on these nodes from the stage's protected reference edges, retaining E_0 and respecting endpoint, ontology, and budget constraints.
  Record the stage graph and check protected state, identity, endpoint, and budget requirements.
end for
Set G_{out} to the final stage graph, or G_0 when no completion stage is enabled.
return G_{out}

Original attributes and reference edges are retained exactly. During a relation-only stage, selected nodes are fixed; during candidate generation, the existing graph is unchanged. These are the operational forms of the contracts in Section 3.1.

Table 19: Checks of protected state and graph validity. Accuracy is assessed separately against reference annotations.

Check family	Required property
Graph well-formedness	Unique stable node identities; finite, bounded boxes; valid node/relation types; existing endpoints; no invalid self-loops or duplicate directed typed edges.
Reference preservation	Protected node attributes/order and reference edges remain unchanged; accepted anchor prefixes remain unchanged.
Relation-only edits	Selected node lists remain exactly fixed; new edges obey typed endpoints, ontology rules, and the declared reference-edge contract.
Node and edge budgets	Per-page node caps and branch/cumulative edge caps are respected; reciprocal units fit atomically within the applicable budget.
Matched-count comparison	Utility reranking and source-score selection retain identical realized per-page node counts and protected anchors.

The GraphDoc audit covers 81 check categories across 1,000 pages and three independently trained pipelines, giving 243,000 recorded checks, all passing. The 243,000 checks repeat 81 categories over stored stages and transitions for the same 1,000 pages and three pipelines.

H End-to-End Runtime Protocol and Resource Accounting

This appendix expands the measured cost comparison in Section 4.5. It covers complete online execution paths and the quality of their actual timed outputs.

Hardware, repeats, and timing boundary. All six systems use the same physical A100-SXM4-40GB GPU, optimizer seed 42, batch size one, and eight-CPU affinity, cores 184–191. Each dataset contributes the same 100 measured pages to every system. Each of three repeats includes 20 warm-up pages per system before these 100 pages. The record therefore contains 1,800 formal page timings and 2,160 output graphs including warm-up. CPU/thread limits are set before numerical-library imports and inherited by workers.

Per-page timing includes image loading, full-page and local visual encoding, pixel/morphology candidates, verification, selection, attachment, CPU/GPU transfers, JSON parsing where applicable, and final graph serialization. No precomputed feature or candidate bank replaces online computation. Model loading is measured separately. Strict JSON Generation uses the structure-only prompt, greedy decoding, and parser in Appendix B. Every actual timed graph is scored, including failures.

Table 20: Complete batch-one runtime and timed-output quality. Mean, median, and p90 are seconds per measured page, retaining all three repeats. Quality is dataset-micro Graph F1 on the actual 100-page outputs.

Dataset / system	Mean	Median	p90	Graph F1
OracleGraph / Guarded Readout	0.8354	0.8137	0.9151	0.392171
OracleGraph / Full	3.5974	3.5687	3.9185	0.395750
OracleGraph / Strict JSON Generation	80.1422	72.2738	96.3806	0.382393
GraphDoc / Base+Local Readout	0.7329	0.6378	1.2108	0.042481
GraphDoc / Matched-count control	0.8488	0.8364	1.1860	0.105963
GraphDoc / Full	0.8773	0.8579	1.2444	0.108323

CUDA allocation and host-memory scope. Table 21 separates PyTorch CUDA allocated memory from host memory. CUDA allocated memory covers PyTorch allocations only, excluding non-PyTorch allocations and host memory. Process-tree RSS is sampled at a requested 0.2-second interval and sums concurrent process RSS, which can double-count shared pages. Shared pages remain counted in each contributing process. The operating-system lifetime high-water mark concerns the main process and can exceed sampled RSS when brief peaks are missed.

Table 21: Memory accounting for the timed execution paths. CUDA is peak allocated MiB; host columns are GiB. RSS and OS high-water mark measure different quantities.

Dataset / system	CUDA	Tree RSS	Main OS HWM
OracleGraph / Guarded Readout	1256.6	3.917	3.917
OracleGraph / Full	1347.1	4.601	3.975
OracleGraph / Strict JSON Generation	17456.0	3.966	7.871
GraphDoc / Base+Local Readout	1512.5	4.255	4.255
GraphDoc / Matched-count control	1512.5	4.384	4.384
GraphDoc / Full	1512.5	4.506	4.506

The OracleGraph Full morphology worker reaches approximately 643 MiB RSS and is included in process-tree accounting. For JSON, the 7.871 GiB OS high-water mark is retained even though the sampled process-tree peak is lower. Full's $1{,}347.1$ MiB and JSON's $17{,}456.0$ MiB CUDA allocations correspond to approximately 1.32 and 17.05 GiB; both numbers cover PyTorch CUDA allocations only.

Cold loading. The three separately recorded OracleGraph model-loading times are $5.55/36.15/38.54$ seconds for Guarded Readout, $39.96/65.68/47.43$ for Full, and $10.38/19.80/16.84$ for JSON. GraphDoc loading across systems is approximately 6.58–8.32 seconds. These separately reported cold-start times vary with shared storage and process initialization; steady-state page latency excludes model loading.