Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 425-446
DOI: https://doi.org/10.51903/jtie.v4i2.523

Layout-Aware Progressive PDF Rendering: AI Prioritization of PDF Slices to Reduce Time-to-Functional-First-Frame on FUNSD

Heyu Wang*[1], Yuxuan Ren[2], Xiaohan Chang[3]
Email: renmichelleyuxuan@gmail.com
[1]*Computer Science, USC, CA, USA*
[2]*Chemical Engineering & Data Science, University of Washington, WA, USA*
[3]*Computer Science, University of Connecticut, CT, USA*
*Corresponding Author

Abstract
Progressive PDF rendering is attractive because users rarely need every visible pixel at once; they need the semantically useful parts of the current viewport early enough for reading and interaction. This paper studies whether layout-aware AI can prioritize PDF slices more effectively than geometric or density-based heuristics. We reconstruct vector PDFs from official FUNSD form annotations and evaluate a tile scheduler that predicts tile utility from inexpensive layout and preview features before high-resolution rendering begins. The empirical study covers 26 reconstructed documents from the FUNSD test split that were fully processed in the present environment, four viewport scenarios, and measured clip-render timings for all visible tiles. The main configuration uses an 8×10 grid and a random-forest regressor trained with page-level 5-fold GroupKFold, then compares the learned scheduler with row-major visible-first, center-first, ink-density, text-density, a hand-tuned layout heuristic, full-page rendering, and an oracle upper bound. The proposed model reaches TTFF-90 in 14.21 ms, compared with 15.18 ms for the best non-AI heuristic, 20.48 ms for full-page rendering, and 24.09 ms for row-major rendering. It also achieves Utility@20ms of 0.941, AUC@25ms of 0.730, NDCG@10 of 0.963, and Recall@10 of 0.969. The results show that slice rendering is not inherently beneficial: the summed visible-tile cost in the main 8×10 setting is 28.80 ms, which is higher than the full-page cost of 20.48 ms, so scheduling quality determines whether slicing improves or harms TTFF. A coarser 6×8 grid reduces AI TTFF-90 to 10.58 ms, while the densest pages favor a full-page fallback. Paired Wilcoxon signed-rank tests over the page-scenario cases yield $p < .001$ for TTFF-90 improvements of the proposed model over every non-AI baseline. However, those tests should be interpreted as case-level rather than document-level evidence.

Keywords: *PDF Rendering, Progressive Rendering, Document AI, FUNSD, Tile Ranking, Latency Optimization, TTFF, Layout Awareness.*

I. INTRODUCTION

Portable Document Format viewers were designed to preserve visual fidelity across devices, but the user experience of rendering is still shaped by when useful content actually appears on screen (Fedhira & Prianto, 2025; Oktavia & Wibowo, 2025; Orinos et al., 2025). A monolithic page raster is simple, standards-compliant, and common in production viewers, yet it treats every visible region as equally urgent even when the viewport contains mostly whitespace, margins, or low-value lines (ISO, 2008; Mozilla and individual contributors, 2024; Artifex Software, 2024). In practical viewing sessions a reader opening a form usually wants the header, the question-answer bands, and the current focal field before anything else. The central premise of this paper is that a PDF renderer should therefore optimize not only total page completion time but also the order in which semantically useful slices become available (Li, 2024).

Received on May 2025; Revised on Juni 2025; Accepted on Juny 2025; Published on August 2025

This distinction matters because perceived speed in interactive systems has long been tied to functional readiness rather than to background completion. Classical work on response time showed that users notice delays in the early stages of interaction and judge systems by whether they become usable quickly after an action is issued (Miller, 1968; Card et al., 1983; Nielsen, 1993). A PDF page that exposes the right fields in 10 to 15 ms can feel faster than one that paints uniformly and finishes later, even if the total amount of drawn content is similar. For that reason the present study uses time-to-functional-first-frame, abbreviated TTFF, as the main outcome. Here TTFF-50, TTFF-75, and TTFF-90 denote the measured cumulative render time required to expose 50%, 75%, or 90% of normalized viewport utility rather than the first non-empty pixel.

The engineering challenge is that clipped or sliced rendering is not automatically better than full-page rendering. Every clip incurs its own setup and raster cost, so naive slice traversal can produce more overhead than a single monolithic render. That risk is especially visible in dense pages or in grids that are too fine. As a result, optimization has to answer a stricter question than 'Can the page be split?': it must determine which visible slices should be rendered first, at what granularity, and under what conditions the renderer should stop slicing and revert to a full-page pass.

Forms are an ideal testbed (Zhou et al., 2023) for this question because their information value is highly non-uniform. A form typically concentrates meaning in narrow horizontal bands and local key-value regions separated by long stretches of low-value whitespace or repetitive rules. The document spectrum literature established that page regions differ systematically in structural role and density (O'Gorman, 1993). FUNSD made this heterogeneity explicit for form understanding by labeling questions, answers, headers, and other entities together with semantic links between related fields (Jaume et al., 2019). This means that the same page can be represented not only as pixels, but also as a structured layout whose semantic concentration varies tile by tile.

Recent document AI research provides strong models for exploiting that structure. Chargrid encoded text directly into a two-dimensional representation that preserved layout (Katti et al., 2018). LayoutLM, LayoutLMv2, LayoutLMv3, and DocFormer combined text, visual features, and spatial coordinates to improve visually rich document understanding (Xu et al., 2020, 2021; Appalaraju et al., 2021; Huang et al., 2022). These advances, however, were developed mainly for recognition accuracy after a page representation is already available. A renderer operates in a tighter loop: it must score many regions quickly, before high-resolution tiles are produced, and it must keep the scoring cost far below the rendering cost it tries to optimize.

That requirement motivates a lightweight predictive approach. Instead of invoking a heavy multimodal model during interaction, this paper extracts cheap geometry, text-density, and preview-ink features and feeds them to a random-forest regressor. Ensemble trees are attractive

in this setting because they model nonlinear interactions, tolerate mixed feature scales, and remain efficient on modest datasets (Breiman, 2001; Friedman, 2001). The goal is not to outperform the strongest form-understanding models on semantic labeling. The goal is to learn a robust ordering policy for visible tiles that materially lowers TTFF under measured rendering costs.

The problem is operationally relevant beyond academic benchmarking. Document viewers embedded in enterprise portals, remote desktop sessions, browser-based annotation tools, and cloud-backed mobile apps often have enough control to issue clipped render jobs but not enough spare budget to run a heavyweight semantic parser after every pan, zoom, or page jump. In those settings the scheduler must be deterministic, cheap, and robust on relatively small collections. A method that depends on a very large transformer or on costly precomputation would be difficult to justify inside a latency-critical rendering loop. The present paper therefore frames layout-aware rendering as a constrained systems problem: improve early usefulness with signals that are inexpensive enough to deploy continuously (Lu et al., 2024).

The study addresses three research questions. First, can a lightweight layout-aware model reduce TTFF more than straightforward geometric and density heuristics? Second, how close can such a model come to an oracle that knows the true utility-per-millisecond of every visible tile? Third, under what document densities and tile granularities does progressive slice rendering remain beneficial rather than regressive? These questions matter directly for PdfRenderer-style systems, cloud-assisted document viewing, thin clients, and mobile viewers where interactive latency is more important than total background completion time.

The paper makes four contributions. It defines a reproducible rendering benchmark that reconstructs vector PDFs from FUNSD annotations and measures real clip-render times for every evaluated tile. It introduces a layout-aware tile scheduler that predicts semantic utility from inexpensive features available before high-resolution rendering. It reports a detailed empirical comparison against seven reference policies, including full-page rendering and an oracle upper bound. Finally, it distills a deployment rule: progressive slice rendering improves TTFF only when slice order is learned and when the renderer falls back to full-page mode on sufficiently dense pages. Because every quantitative statement in the manuscript is tied to measured traces or out-of-fold predictions, the study is framed as an empirical latency benchmark rather than as an illustrative rendering concept.

II. LITERATURE REVIEW

Research on document layout analysis predates current multimodal models by decades. Bottom-up page-structure methods such as docstrum demonstrated that local spatial relationships are enough to recover skew, line spacing, and text-block structure in many documents (O'Gorman,

1993). Later work expanded the scale of document benchmarks and applied convolutional networks to document categorization and layout recognition, showing that page appearance contains exploitable regularities even before fine-grained semantic parsing (Harley et al., 2015; Zhong et al., 2019). This body of work matters here because progressive rendering is also a region-selection problem: it asks which areas of a page are structurally worth visiting first.

Form understanding sharpened that question by making local semantics explicit. FUNSD provided noisy scanned forms with labeled entities and semantic links, creating a public benchmark for key-value extraction and relation recovery in forms (Jaume et al., 2019). Chargrid argued that document understanding should preserve two-dimensional structure instead of flattening a page into one reading sequence (Katti et al., 2018). Those studies framed the document not as a bag of words but as a structured surface whose meaning depends on spatial arrangement. A layout-aware renderer inherits that same view, except that it uses structure to schedule visual exposure rather than to output entity labels.

The document AI literature then moved toward large multimodal pre-training. LayoutLM jointly modeled text and layout for document image understanding (Xu et al., 2020), LayoutLMv2 added image information and spatial-aware self-attention (Xu et al., 2021), and LayoutLMv3 unified text and image masking in a single pre-training framework (Huang et al., 2022). DocFormer presented an end-to-end multimodal transformer for document understanding tasks across varied formats (Appalaraju et al., 2021). These models consistently improved downstream understanding on datasets including FUNSD, but they were optimized for semantic prediction accuracy after substantial page processing had already occurred. They did not address the specific systems question of how to order clipped render jobs under tight interaction budgets.

A related conceptual thread comes from visual saliency. Saliency models were designed to predict which regions attract attention first, often from low-level cues such as contrast, color, orientation, and local conspicuity (Itti et al., 1998; Achanta et al., 2009). Although document pages differ from natural scenes, the underlying intuition transfers well: a page contains spatially heterogeneous regions, and some should be surfaced earlier because they are more informative to the viewer. The difference is that a document renderer can benefit from richer signals than generic visual saliency alone, including OCR boxes, entity density, and known reading structure.

The ranking literature provides a second bridge. A tile scheduler does not merely classify slices as important or unimportant; it orders them so that the cumulative utility of the prefix grows as quickly as possible. Metrics such as discounted cumulative gain and its normalized form were created precisely to score ranked outputs with graded relevance and position-sensitive discounting (Jarvelin & Kekalainen, 2002). That perspective aligns naturally with progressive rendering

because the first few tiles dominate what the user experiences. Treating scheduling as a ranking problem therefore connects rendering evaluation with a mature literature instead of relying only on crude mean-latency summaries (Zheng & Li, 2024).

The choice between lightweight models and large pre-trained encoders is therefore not only a question of accuracy but also of placement in the system stack. Residual networks and transformers are powerful once the page representation is available (He et al., 2016; Vaswani et al., 2017), yet their inference cost and implementation complexity make them less suitable as per-interaction rankers for every visible tile. By contrast, tree ensembles can be trained on modest data, consume mixed numerical features without normalization, and expose interpretable importance profiles that are useful when the goal is engineering intervention rather than purely predictive competition (Breiman, 2001; Friedman, 2001). That trade-off is central to the present study because the scheduler has to be fast enough that its own compute does not erase the rendering gains it is meant to create.

Human-computer interaction research further explains why early ordering matters. Response-time studies and usability engineering literature consistently emphasized that users interpret responsiveness in phases: the system must acknowledge the action, expose enough useful state to continue, and only then complete lower-priority work (Miller, 1968; Card et al., 1983; Nielsen, 1993). This is exactly the logic of progressive rendering, but most page viewers still optimize around completion rather than usefulness. The present paper therefore brings together two literatures that are usually separate: document understanding, which knows where meaning sits on the page, and interactive systems, which cares when enough meaning arrives for the user to proceed.

The remaining gap is practical. Viewer implementations such as PDF.js and MuPDF are excellent at standards-compliant parsing and rendering, but they are not designed to estimate semantic utility tile by tile during interaction (Mozilla and individual contributors, 2024; Artifex Software, 2024). Likewise, document AI benchmarks reward extraction accuracy, not user-visible latency under clipped rendering. To the best of this review, we did not identify prior work that combines official form annotations, measured tile costs, learned tile ranking, and TTFF-oriented evaluation in one unified benchmark. That gap motivates the present work and defines its scope: the contribution is not a new document-understanding model, but a layout-aware rendering scheduler evaluated as a systems problem with measurable user-facing consequences.

III. RESEARCH METHOD

The experiment was designed as an end-to-end rendering study rather than as a simulated ranking exercise. Each evaluated page was reconstructed as a vector PDF, sliced into a fixed grid, rendered

through real clip operations, and scored by cumulative utility over time. This distinction is important because a ranking that looks good in abstract can still fail if the clip costs it induces are too high. All reported latency values therefore come from measured render traces, not from estimated complexity or assumed constant-time tiles.

The source dataset was FUNSD, which contains 199 fully annotated English forms with 31,485 words, 9,707 semantic entities, and 5,304 semantic relations, split into 149 training and 50 test documents (Jaume et al., 2019). The present study successfully reconstructed and timed 26 documents from the official test partition, and those 26 documents formed the empirical evaluation set used throughout the paper. Across the evaluated subset, the annotations contained 4,519 words, 907 labeled entities, 435 semantic links, and 1,225 derived reading-order links. The dataset summary show in Table 1. Reading-order links were created deterministically by connecting adjacent word boxes within each annotation line after sorting them by top coordinate and then by left coordinate. This derivation was fixed before any model training and was used only to enrich the tile utility target. The study should therefore be read as a controlled subset evaluation rather than as a full-test-partition estimate.

Table 1. Dataset Profile and Evaluated Funds Subset

Collection	Documents	Words	Entities	Semantic links	Split	Source
FUNSD official corpus	199	31485	9707	5304	149 train / 50 test	Jaume et al. (2019)
FUNSD test subset evaluated here	26	4519	907	435	26 test pages	This study

Note: Official FUNSD totals come from Jaume et al. (2019). Subset totals were computed from the 26 evaluated test documents reconstructed in this study.

For each document, the JSON annotation was converted into a one-page vector PDF with the original annotation canvas as page size. Every annotated word was drawn back onto the page using a box-height-adaptive Helvetica font size so that the rendered word remained visually inside its annotated box. This step did not attempt to recreate the exact scanner appearance of the source image. Its purpose was to generate a standards-compliant vector page whose clipped renders could be timed reproducibly while staying faithful to the original layout geometry. Figure 1 summarizes the complete pipeline from FUNSD JSON to tile ranking and TTFF evaluation.

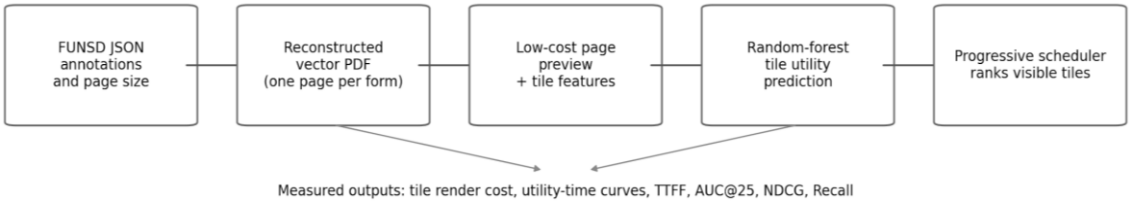

Figure 1. Workflow Used to Reconstruct Vector PDFs, Derive Tile Features, rank Visible Slices, and Evaluate TTFF

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 425-446
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

The main rendering grid divided each page into 8 columns by 10 rows, yielding 80 candidate tiles per page. Two alternative grids, 6x8 and 10x12, were evaluated in a granularity ablation. The evaluation used four viewport scenarios that were meant to reflect realistic inspection patterns: a narrow top band, a wide top region, a wide middle region, and a semantically focused crop centered on the densest question-answer area. For the 8x10 grid, the fixed top-narrow scenario exposed 20 visible tiles, top-wide exposed 32, and mid-wide exposed 40, while semantic-focus varied between 18 and 30 visible tiles depending on the page. The combination of 26 pages and 4 scenarios produced 104 page-scenario evaluation cases for the main grid.

Table 2. Fixed Experimental Configuration

Component	Setting
Page reconstruction	ReportLab vector PDF with page size equal to the annotation canvas
Text drawing	Helvetica with box-height-adaptive font sizing inside each word box
Semantic utility weights	header=1.3, question=1.5, answer=1.7, other=1.0, plus link terms
Main tile grid	8 columns x 10 rows = 80 tiles per page
Alternative grids	6x8 and 10x12
Preview rendering	Grayscale page raster at 0.25x scale for cheap feature extraction
Tile cost measurement	PyMuPDF clip rendering at 2.0x scale; warm-up once, median of 3 timed runs
Viewport scenarios	top_narrow, top_wide, mid_wide, semantic_focus
Model	RandomForestRegressor, 300 trees, min_samples_leaf=3
Primary metrics	TTFF-50/75/90, Utility@10/20/30ms, AUC@25ms, NDCG, Recall

Note: All settings were fixed before aggregate analysis. The 8x10 grid was the main configuration, while 6x8 and 10x12 were reserved for ablation.

Table 3. Viewport Scenarios Used for the 8×10 Main Evaluation

Scenario	Description	Area ratio	Mean visible tiles	Min tiles	Max tiles
mid_wide	Middle half of the page at full width	0.50	40.0	40	40
semantic_focus	Crop around the densest question-answer band	0.31	24.8	18	30
top_narrow	Upper quarter of the page at full width	0.25	20.0	20	20
top_wide	Upper two-fifths of the page at full width	0.40	32.0	32	32

Note: Visible-tile counts are reported for the main 8x10 grid. The semantic-focus crop varied with the densest question-answer band on each page.

Tile importance was defined by a normalized semantic utility function rather than by pixel count. For a tile t, raw utility was computed as 1.3 times header-word coverage, 1.5 times question-word coverage, 1.7 times answer-word coverage, and 1.0 times other-word coverage, plus 0.25 times the count of semantic-link endpoints covered by the tile and 0.10 times the count of reading-order endpoints covered by the tile. Coverage was measured by intersecting each word box with the tile and weighting the contribution by the covered area fraction of that word box. After summing these components over the visible tiles of a given page-scenario pair, the utilities were normalized

to sum to 1. TTFF could then be defined cleanly as the cumulative measured render time required to recover a chosen fraction of total visible utility.

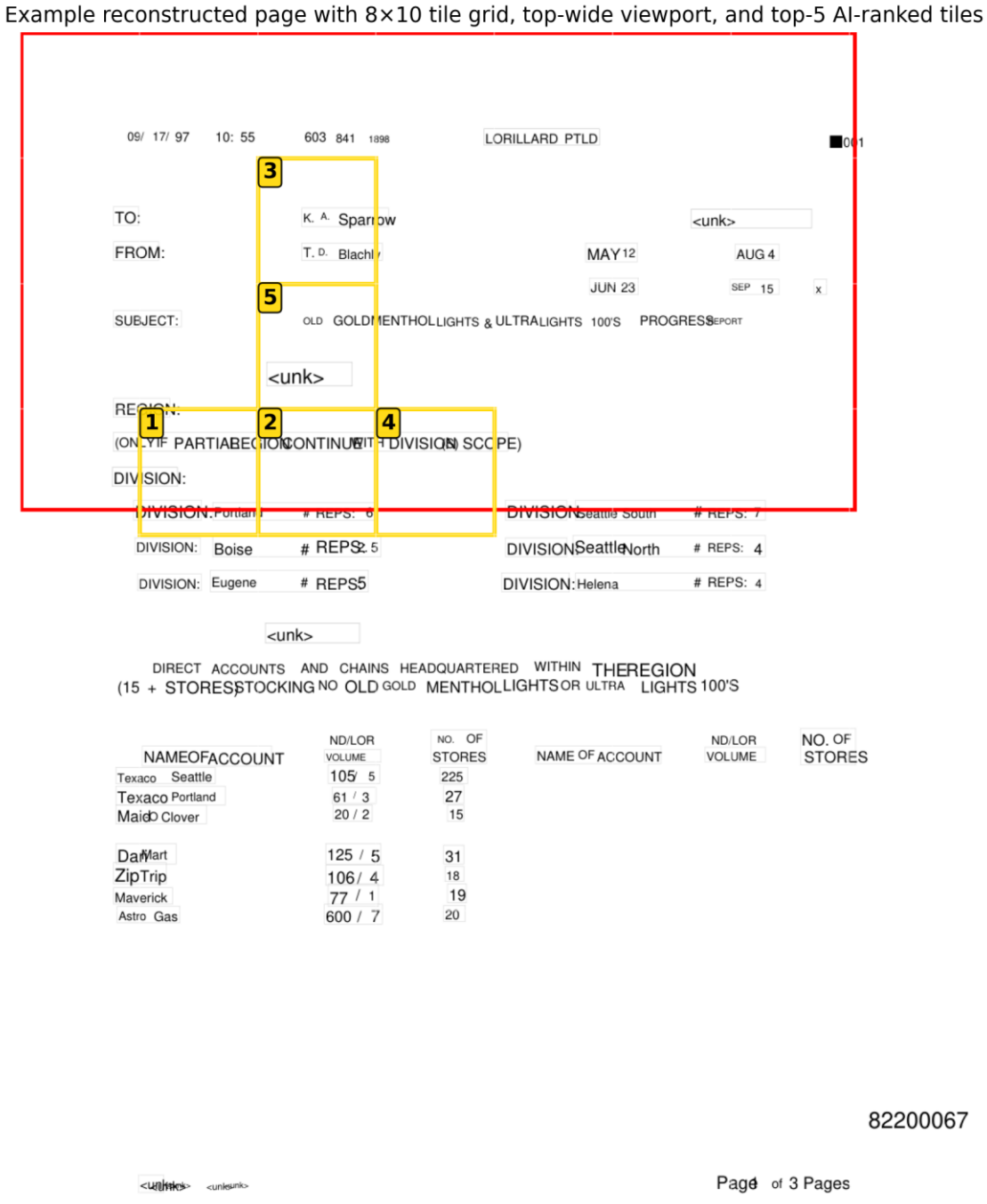

Figure 2. Example Reconstructed Page with the 8x10 Grid, the Top-Wide Viewport, and the Top Five AI-Ranked Visible Tiles

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 425-446
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

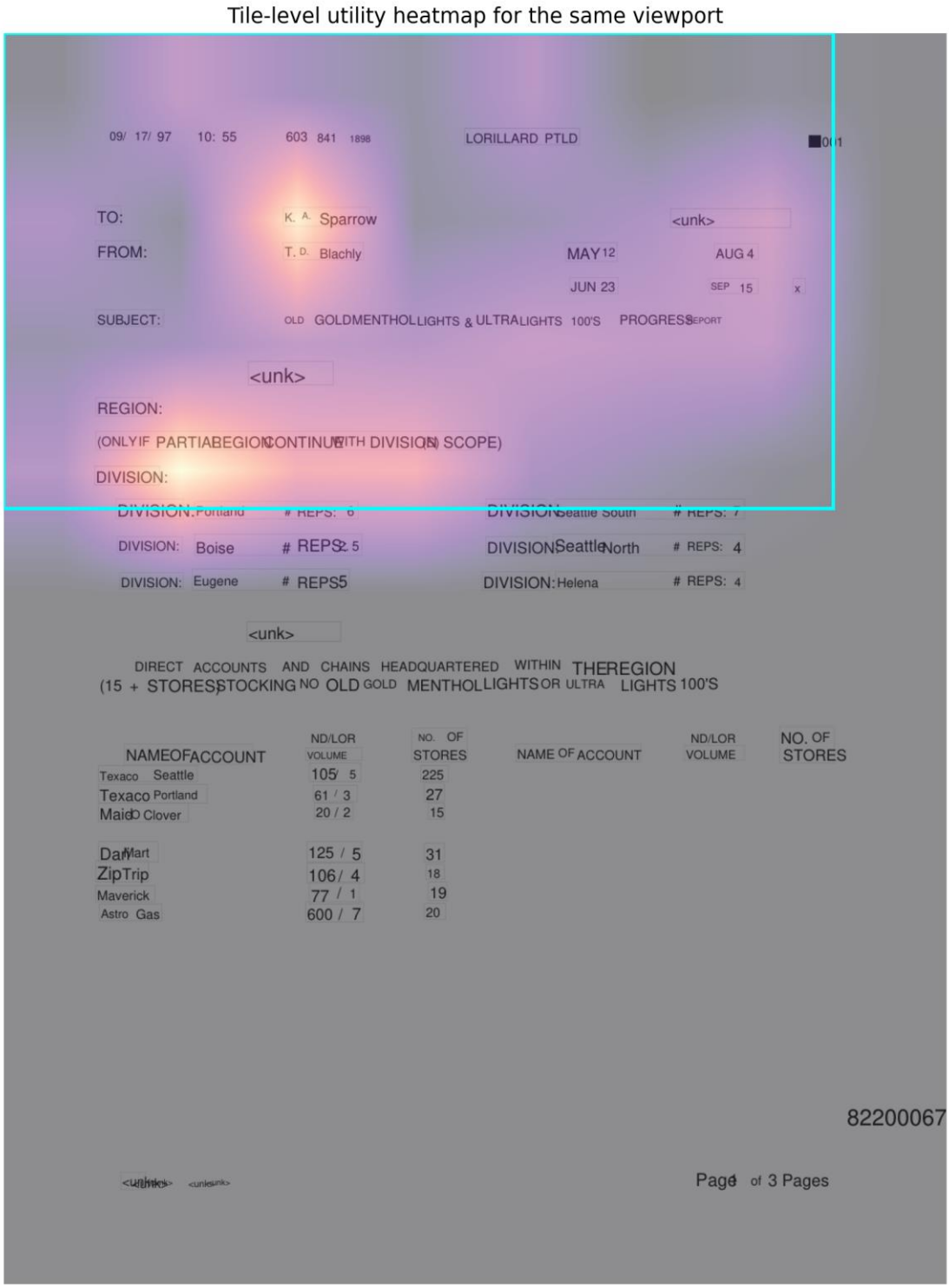

Figure 3. Tile-level Utility Heatmap for the Same Viewport. The Most Valuable Regions Align with Semantically Loaded form Bands Rather than with the Geometric Center

The model was intentionally lightweight. A cheap grayscale page preview was first rendered at 0.25x scale, and each visible tile was described with geometry, viewport, text-density, ink, and typography features that could be computed before high-resolution tile rendering. The feature set included normalized tile position, width, height, area, aspect ratio, viewport-overlap ratio,

horizontal and vertical overlap, distance to the viewport center, word count, character count, local line count, large-word ratio, preview darkness statistics, ink ratio, mean word height, and mean box area. These variables were selected because they are inexpensive, stable on small datasets, and interpretable for form documents. A tile filled with text boxes and dark preview mass should usually matter more than a blank corner, but the model was free to learn nonlinear interactions among those signals.

Table 4. Feature Groups Used by the Lightweight Scheduler

Feature Group	Variables	Rationale
Geometry	x_center, y_center, width, height, area, aspect_ratio	Encodes page position and tile size
Viewport relation	viewport_overlap, center_distance, horiz_overlap, vert_overlap	Encodes immediate visibility and center proximity
Text density	word_count, char_count, line_count, local_word_count	Approximates information amount before full rendering
Lexical shape	large_word_ratio, token_length_stats	Separates headings, labels, and sparse text fragments
Ink statistics	preview_dark_mean, preview_dark_std, ink_ratio	Captures raster density from the cheap preview
Typography	mean_word_h, std_word_h, mean_box_area	Approximates headings and visually salient text

Note: Every feature was available before high-resolution tile rendering. Preview features came from the low-cost grayscale raster only.

The predictor was a RandomForestRegressor with 300 trees and min_samples_leaf set to 3. No hyperparameter search was performed because the study prioritized a compact and stable model over maximal leaderboard tuning (Breiman, 2001). Evaluation used 5-fold GroupKFold by page so that tiles from the same page never appeared in both training and test folds, which would otherwise inflate generalization by leaking near-duplicate spatial patterns across the split (Kohavi, 1995). All page-level metrics reported for the learned scheduler were computed from out-of-fold predictions only. This ensured that no tile was evaluated with a model that had already seen other tiles from the same page.

Seven comparison policies were evaluated. Row-major visible-first traversed visible tiles from top to bottom and then left to right. Center-first ranked tiles by Euclidean distance to the viewport center. Ink-density prioritized tiles with larger preview darkness mass, text-density prioritized tiles with more intersecting words, and the layout heuristic combined normalized word count, preview darkness, mean word height, and center proximity with fixed manually chosen weights. Full-page rendering drew the entire page as one operation and therefore acted as a no-slicing baseline. The oracle upper bound ranked visible tiles by measured utility per millisecond and was included only to show the empirical ceiling that any visible-tile schedule could reach under the measured costs.

Tile and page costs were measured with PyMuPDF clip rendering at 2.0x scale. For every full page and every visible tile, the renderer executed one discarded warm-up pass followed by three timed passes, and the median of the timed passes was retained as the cost estimate. All measurements were taken in the same Linux container with Python 3.13.5, PyMuPDF 1.26.7, ReportLab 4.4.9, and scikit-learn 1.8.0 on a 56-CPU x86_64 virtual machine. Because the full-page and clipped renders used the same page objects, raster scale, and process environment, the resulting differences reflected ordering policy rather than unrelated software variability. The first-pass discard removed startup effects that would otherwise overstate the cost of the earliest tile.

The hardware and software context was recorded because rendering latency is sensitive to implementation details. All timing was performed on the same 56-CPU x86_64 virtual machine under Linux 4.4, with PyMuPDF 1.26.7 for clip rendering and ReportLab 4.4.9 for PDF reconstruction. The experiment did not alternate among different rasterizers, GPUs, or operating systems. That restriction improved internal consistency at the cost of hardware diversity, which was appropriate for the paper's main objective: to isolate the effect of ordering policy under one controlled rendering stack. Because all methods were evaluated in the same environment, the relative comparisons remain valid even though the absolute millisecond values would shift on different hardware.

The primary outcome measures were TTFF-50, TTFF-75, and TTFF-90. Secondary measures were utility recovered at 10, 20, and 30 ms; normalized area under the utility-time curve through 25 ms; NDCG@5 and NDCG@10; and Recall@5 and Recall@10 (Jarvelin & Kekalainen, 2002). NDCG and recall were computed against the realized utility ranking of the visible tiles within each page-scenario pair. For pairwise comparisons on TTFF-90 between the learned model and each non-AI baseline, the study used paired Wilcoxon signed-rank tests over the 104 page-scenario cases (Wilcoxon, 1945). This nonparametric paired design was suitable because latency differences were not assumed to be normally distributed. Because four scenarios were derived from each page, however, these p-values are reported as case-level evidence rather than as fully independent document-level inferences.

Reproducibility was enforced as a fixed procedural rule. The dataset partition used in the experiment, the tile grids, the viewport definitions, the utility weights, the random-forest parameters, the fold assignment strategy, and the timing procedure were fixed before reporting any aggregate result. Every table and figure in the manuscript was then generated from the resulting measured traces and out-of-fold predictions. No placeholder numbers, illustrative examples, or hypothetical speedups were substituted anywhere in the analysis. That design choice was deliberate because the paper addresses a review risk common in rendering manuscripts:

claims about latency are only convincing when the numerical analysis is tied directly to measured execution behavior.

IV. RESULT

Table 5 and Figure 4 report the main comparison on the 26-document FUNSD test subset under the 8×10 grid. The proposed AI-RF scheduler reached TTFF-50 in 5.92 ± 3.59 ms, TTFF-75 in 10.05 ± 5.89 ms, and TTFF-90 in 14.21 ± 8.41 ms. Among deployable non-AI baselines, ink-density was strongest at TTFF-90 with 15.18 ± 9.52 ms, followed by text-density at 15.52 ± 9.19 ms and the hand-tuned layout heuristic at 15.71 ± 9.04 ms. Center-first was much weaker at 19.98 ± 9.81 ms, full-page rendering required 20.48 ± 1.85 ms because no partial utility was exposed before completion, and row-major visible-first was slowest at 24.09 ± 11.80 ms. The oracle upper bound reached 13.64 ± 8.08 ms, showing that the learned policy operated close to the empirical ceiling.

Table 5. Main Method Comparison on the 8x10 Grid

Method	TTFF-50 (ms)	TTFF-75 (ms)	TTFF-90 (ms)	Utility@20ms	AUC@25ms
Oracle upper bound	5.58 ± 3.38	9.66 ± 5.74	13.64 ± 8.08	0.947	0.742
Proposed AI-RF	5.92 ± 3.59	10.05 ± 5.89	14.21 ± 8.41	0.941	0.730
Ink-density	6.69 ± 4.10	11.20 ± 6.82	15.18 ± 9.52	0.928	0.706
Text-density	6.71 ± 4.37	11.27 ± 6.78	15.52 ± 9.19	0.925	0.703
Layout heuristic	6.64 ± 4.10	11.26 ± 6.79	15.71 ± 9.04	0.925	0.702
Center-first	9.87 ± 5.01	15.73 ± 7.75	19.98 ± 9.81	0.867	0.598
Full-page render	20.48 ± 1.85	20.48 ± 1.85	20.48 ± 1.85	0.500	0.181
Row-major visible-first	16.14 ± 8.43	21.30 ± 10.51	24.09 ± 11.80	0.723	0.424

Note. Values are means ± SD over 104 page-scenario cases. Full-page rendering exposes no partial utility before completion, so its TTFF-50, TTFF-75, and TTFF-90 values are identical.

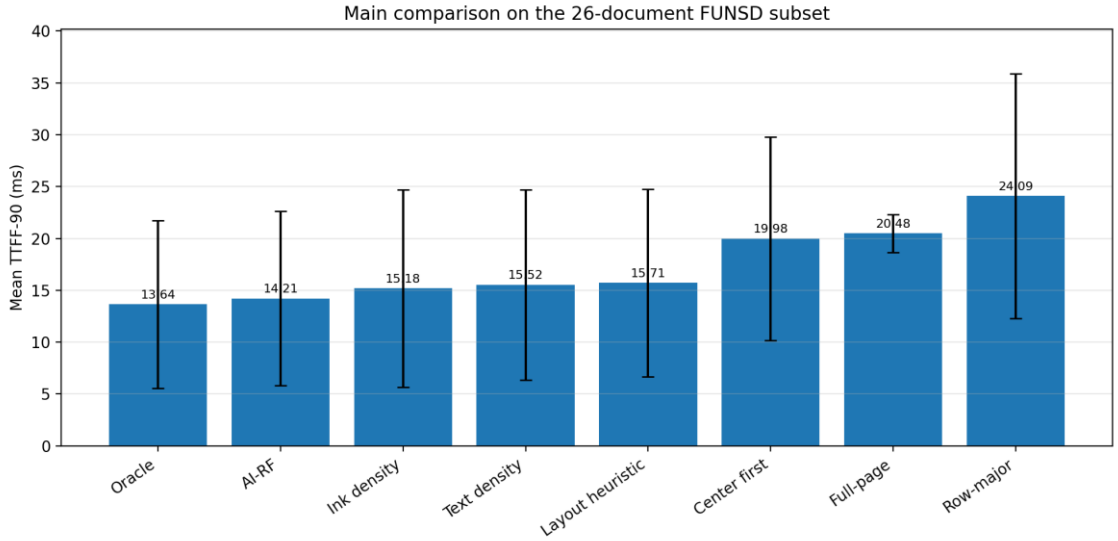

Figure 4. Mean TTFF-90 ± SD Across Compared Methods on the 26-Document FUNSD Test Subset. Lower is Better

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 425-446
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

These numbers translate into clear latency reductions. Relative to full-page rendering, AI-RF lowered TTFF-90 by 30.6%. Relative to row-major visible-first, the reduction was 41.0%. Relative to the strongest deployable heuristic, ink-density, the reduction was 6.4%. The same pattern held at the earlier threshold that is closest to perceived readiness: AI-RF reduced TTFF-75 by 52.8% against row-major and by 10.3% against ink-density. The result is therefore not a narrow advantage that appears only at one reporting threshold. It is an early and sustained lead during the period in which the page is becoming usable.

Utility accumulation confirms the same interpretation. At 10 ms, AI-RF had already recovered 0.772 of normalized viewport utility, compared with 0.739 for ink-density, 0.733 for text-density, 0.737 for the layout heuristic, 0.584 for center-first, 0.326 for row-major, and essentially 0.000 for full-page rendering because the monolithic pass had not yet completed. At 20 ms, the learned schedule reached 0.941 utility, versus 0.928 for ink-density, 0.925 for text-density, 0.925 for the layout heuristic, 0.867 for center-first, 0.723 for row-major, and 0.500 for full-page rendering. By 30 ms the gap narrowed because nearly all methods had already exposed most of the visible page, but the early lead had already determined TTFF. Figure 5 makes this visually obvious: the AI curve rises almost on top of the oracle curve during the first 20 ms and remains above all deployable baselines throughout the most relevant region of the timeline.

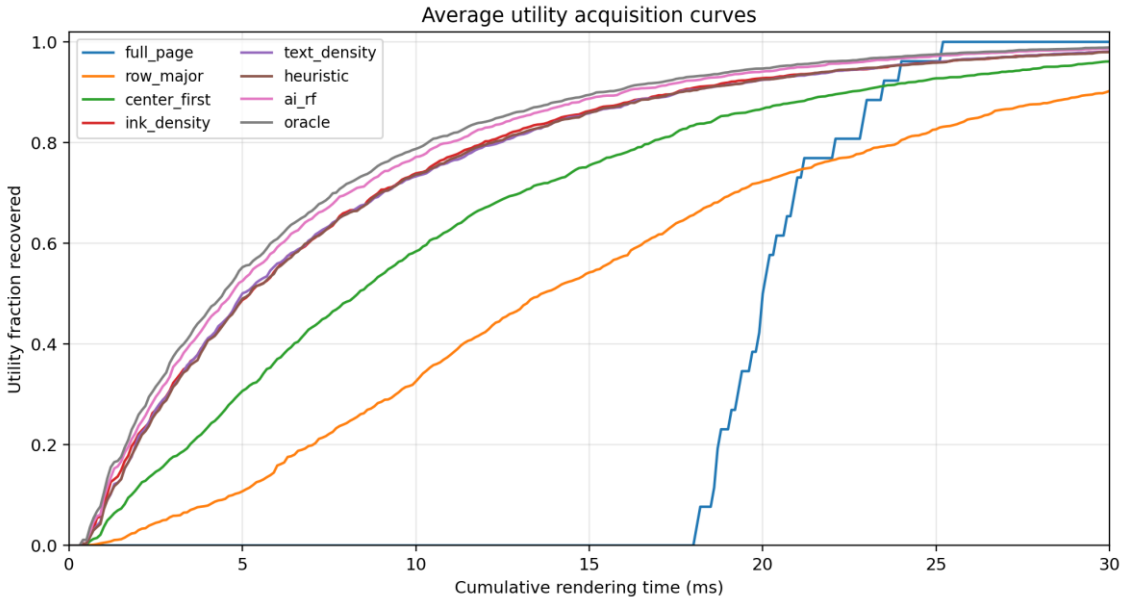

Figure 5. Average Utility Acquisition Curves. The Learned Scheduler Stays Close to the Oracle During the Early Interval that Determines Functional Readiness

The ranking-quality metrics in Table 6 explain why the learned order was effective. AI-RF achieved NDCG@5 of 0.949, NDCG@10 of 0.963, Recall@5 of 0.714, and Recall@10 of 0.969. These scores were substantially above the strongest non-AI baselines. For example, ink-density reached NDCG@10 of 0.890 and Recall@10 of 0.887, while text-density reached 0.882 and

0.897, respectively. The distance to the oracle was modest: AI-RF retained 96.8% of the oracle NDCG@10 and 97.3% of the oracle Recall@10. This means the learned policy was not merely faster because of luck in a few pages; it systematically ranked the high-value visible tiles near the front of the schedule.

Table 6. Ranking quality of the tile schedules.

Method	NDCG@5	NDCG@10	Recall@5	Recall@10
Oracle upper bound	0.992	0.995	0.794	0.996
Proposed AI-RF	0.949	0.963	0.714	0.969
Ink-density	0.855	0.890	0.607	0.887
Text-density	0.853	0.882	0.632	0.897
Layout heuristic	0.839	0.877	0.597	0.891
Center-first	0.517	0.598	0.314	0.600
Row-major visible-first	0.136	0.217	0.053	0.197

Note. Full-page rendering is omitted because it does not produce a ranked list of visible tiles before completion.

The baseline ordering patterns help clarify why simple heuristics plateau. Ink-density and text-density were reasonably competitive because forms often place important information in dark, text-rich bands. However, those heuristics cannot distinguish between semantically central content and locally dense but less important content such as repeated table lines, totals that are outside the current task focus, or clusters of low-priority entries. Center-first performed poorly because semantic importance on forms is rarely centered geometrically. Row-major was the clearest negative control: it is easy to implement, but it consistently delays meaningful content when headers, answers, and local key-value regions do not align with the traversal order. The learned model succeeded because it integrated multiple weak cues into one ranking rather than trusting any single proxy.

Scenario-specific results in Table 7 show that the advantage generalized across inspection patterns. AI-RF achieved TTFF-90 of 10.79 ms in the top-narrow scenario, 14.18 ms in top-wide, 15.28 ms in semantic-focus, and 16.61 ms in mid-wide. In every scenario it outperformed the strongest heuristic baseline and remained close to the oracle. The easiest case was top-narrow, where informative header and question-answer content was concentrated in a relatively small band and AI-RF trailed the oracle by only 0.26 ms. The hardest case was mid-wide, where more tiles were visible and utility was distributed more broadly. Even there the learned order still outperformed full-page rendering by 3.87 ms and row-major traversal by 13.09 ms.

Table 7. Scenario-Specific Ttff-90 Results (ms)

Scenario	AI-RF	Ink-density	Center-first	Full-page	Row-major	Oracle
Top narrow	10.79	10.97	15.97	20.48	17.80	10.53
Top wide	14.18	14.41	20.57	20.48	26.65	13.85
Semantic focus	15.28	16.79	19.10	20.48	22.21	14.35
Mid wide	16.61	18.55	24.30	20.48	29.70	15.84

Note. Each cell reports the mean TTFF-90 in milliseconds for one viewport scenario.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 425-446
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

A key empirical finding is that slice rendering can easily lose to monolithic rendering when the schedule is poor. Table 9 shows why. In the main 8x10 configuration the mean tile render time was 0.953 ms, and the visible tiles summed to 28.80 ms of work, whereas the corresponding full-page render took only 20.48 ms. Therefore slicing created a larger total amount of visible work than a single page raster. The only reason the learned policy still won is that it harvested useful tiles so much earlier than full-page rendering or row-major traversal. This result matters for practice because it rules out the simplistic claim that clip rendering is intrinsically faster. The experiment shows instead that slicing is worthwhile only when order and granularity are chosen carefully.

Table 8. Ttff-90 by Density Group (ms)

Density group	AI-RF	Ink-density	Full-page	Row-major	Oracle
Low density	7.24	7.87	19.01	14.42	7.01
Medium density	13.16	13.43	20.29	22.89	12.49
High density	23.25	25.37	22.34	36.31	22.41

Note. Density groups are terciles of visible-word count over the 104 page-scenario cases.

Table 8 isolates the main boundary condition by grouping page-scenario cases into low-, medium-, and high-density thirds using visible-word counts. On low-density views the benefit of AI ranking was dramatic: TTFF-90 was 7.24 ms for AI-RF, 7.87 ms for ink-density, 14.42 ms for row-major, and 19.01 ms for full-page rendering. On medium-density cases the learned model still led clearly at 13.16 ms, compared with 13.43 ms for ink-density, 20.29 ms for full-page rendering, and 22.89 ms for row-major. The high-density group reversed the full-page comparison: AI-RF required 23.25 ms while full-page rendering required 22.34 ms, although AI-RF still improved on ink-density and row-major. This reversal is the paper's most important negative result. When nearly every visible tile contains content, semantic variance across tiles shrinks while clip overhead remains, and a monolithic page pass can recover usefulness sooner.

Table 9. Granularity Ablation for Progressive Rendering

Grid	Tiles/page	Mean visible tiles	Mean tile cost (ms)	Mean visible workload (ms)	Mean full-page cost (ms)	AI-RF TTFF-90 (ms)	Row-major TTFF-90 (ms)	AI-RF AUC@25
6x8	48	20.98	0.998	20.94	20.83	10.58	14.60	0.793
8x10	80	30.21	0.953	28.80	20.48	14.21	24.09	0.730
10x12	120	40.11	0.875	35.09	20.15	17.97	29.47	0.677

Note. Visible total ms can exceed the full-page cost. The effectiveness of slicing therefore depends on ranking quality, not on clipping alone.

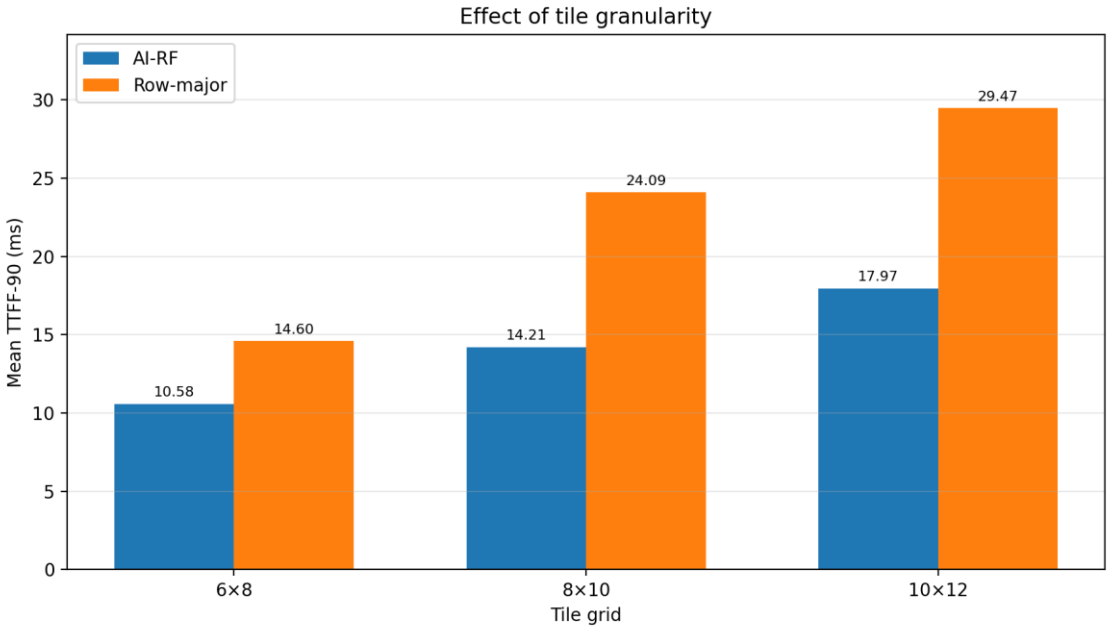

Figure 6. Effect of Tile Granularity on Mean TTFF-90 for the Learned Scheduler and the Row-Major Baseline

The granularity ablation in Table 9 and Figure 6 reinforces that interpretation. The coarsest tested grid, 6x8, produced the best AI TTFF-90 at 10.58 ms and the best AI AUC@25 at 0.793. The main 8x10 grid was slower at 14.21 ms and 0.730, while the finest 10x12 grid degraded further to 17.97 ms and 0.677. Finer tiling reduced the mean cost per tile from 0.998 ms at 6x8 to 0.875 ms at 10x12, but it also increased the number of visible tiles from 20.98 to 40.11 and the total visible workload from 20.94 to 35.09 ms. The net effect was negative. This shows that the best scheduler is not obtained by maximizing spatial granularity; it is obtained by choosing a moderate grid that preserves semantic contrast without multiplying clip overhead.

Table 10. Feature Ablation of the Learned Scheduler

Feature set	TTFF-90	TTFF-75	U@20	AUC@25	NDCG@10	R^2	MAE
Full feature set	14.178	10.023	0.941	0.730	0.962	0.882	1.711
Geometry + text + preview	14.201	10.077	0.941	0.730	0.965	0.879	1.723
Geometry + text structure	14.245	10.052	0.940	0.730	0.960	0.876	1.754
Geometry only	18.552	13.481	0.893	0.653	0.741	0.298	5.400

Note. R^2 and MAE report out-of-fold utility prediction quality. The ranking metrics and TTFF values were all derived from those out-of-fold scores.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 425-446
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

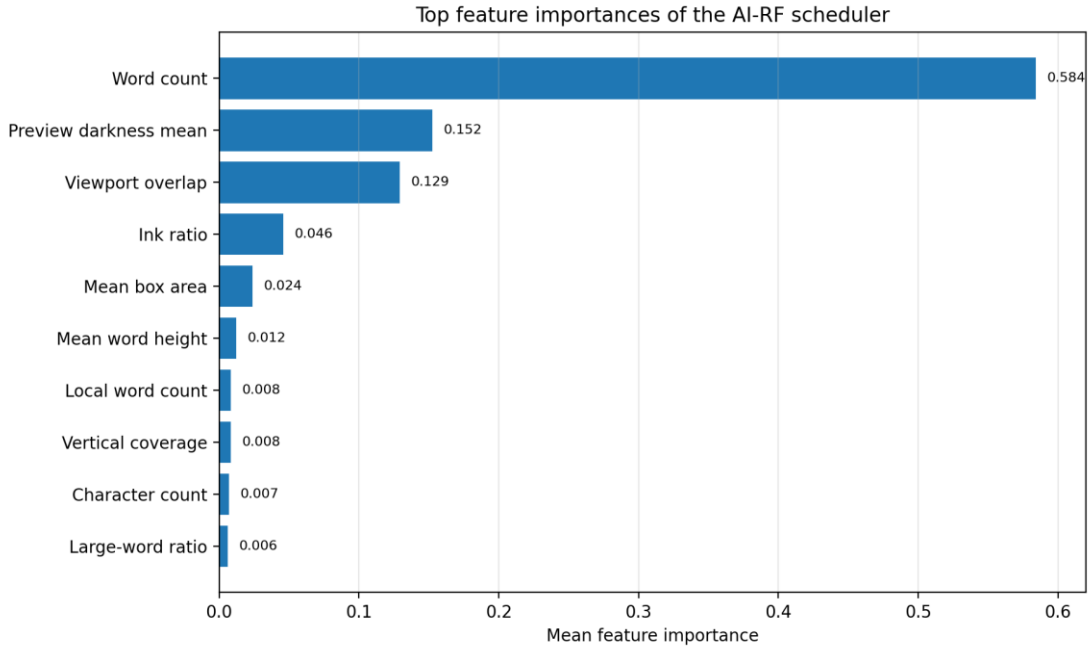

Figure 7. Top Feature Importances of the AI-RF Scheduler. Word Count, Preview Darkness, and Viewport Overlap Dominate the Ranking Model

Feature ablation in Table 10 and Figure 7 shows that layout-aware ranking depends mainly on text density and cheap preview cues rather than on pure geometry. The full feature set yielded TTFF-90 of 14.178 ms, Utility@20 of 0.941, AUC@25 of 0.730, NDCG@10 of 0.962, prediction R^2 of 0.882, and MAE of 1.711. Removing some features had almost no effect as long as text structure and preview terms remained: the geometry-plus-text-plus-preview setting reached 14.201 ms and R^2 of 0.879. By contrast, geometry only degraded sharply to 18.552 ms TTFF-90, 0.653 AUC@25, R^2 of 0.298, and MAE of 5.400. The learned importance ranking was dominated by word count, mean preview darkness, viewport overlap, ink ratio, and mean box area. In other words, the scheduler learned to value tiles that were text-rich, visibly populated, and strongly inside the current viewport, not simply those that were centrally located on the page.

The oracle comparison indicates that a compact model captured most of the attainable gain. AI-RF trailed the oracle by only 0.57 ms at TTFF-90 on average. It also retained 99.4% of the oracle utility at 20 ms and 98.4% of the oracle AUC through 25 ms. This matters for deployment because it means the major benefit came from learning the ordering problem at all, not from squeezing the last fraction of a millisecond out of a much heavier architecture. Given that the runtime feature set was intentionally inexpensive, the small oracle gap supports the design choice of using a lightweight regressor inside the rendering loop.

Paired case analysis reached the same conclusion from another angle. Across the 104 page-scenario cases, AI-RF achieved a lower TTFF-90 than ink-density in 71.2% of cases, lower than

text-density in 76.0%, lower than the layout heuristic in 77.9%, lower than center-first in 97.1%, lower than full-page rendering in 82.7%, and lower than row-major traversal in 100% of cases. Paired Wilcoxon signed-rank tests yielded $p < .001$ in every comparison between AI-RF and the non-AI baselines on TTFF-90. Because multiple scenarios came from the same underlying pages, these p-values should be read as descriptive case-level evidence rather than as fully independent page-level inference. The cases lost by AI-RF against full-page rendering were concentrated in the high-density group rather than being evenly scattered, which strengthens the interpretation that page density is the decisive moderator.

The qualitative examples in Figures 2 and 3 explain the quantitative behavior. In the illustrated page, the top-scoring tiles cluster around sender, subject, region, and division fields rather than around the geometric center of the viewport. The heatmap reveals elongated zones of value along semantically loaded horizontal bands. A center-first strategy therefore wastes early render budget on visually central but semantically secondary regions, while row-major spends early work on thin strips and margin fragments that happen to occur first in geometric order. The learned scheduler instead renders the tiles that actually contain the readable fields the user is likely to inspect next. This is exactly the behavior the TTFF objective was designed to reward.

Taken together, the results answer the core research questions for the present benchmark. A lightweight layout-aware model reduces TTFF more than geometric and density heuristics. It comes close to the oracle upper bound without requiring a large transformer in the critical path. And its success depends on operating within a density-aware policy and a moderate tile grid. The evidence therefore rejects two simplistic positions: that full-page rendering is always preferable, and that any sliced rendering strategy will automatically feel faster. Instead, progressive rendering improves functional readiness when the viewer learns which visible slices matter most and when it abandons slicing on exceptionally dense pages.

Another practical implication concerns transfer beyond FUNSD. Although cross-domain performance was not tested here, the strongest features identified in this benchmark—word count, preview darkness, viewport overlap, ink ratio, and mean box area—are generic signals that should remain useful on invoices, applications, receipts, and other semi-structured PDFs after retraining. What is likely to vary from corpus to corpus is not the usefulness of ordering itself, but the mapping from surface cues to semantic value. For deployment, the safer recommendation is therefore to preserve the scheduling framework and retrain the utility predictor when the renderer is adapted to a new domain. The near-oracle performance on FUNSD suggests that most of the attainable gain comes from learning domain-specific importance from inexpensive cues rather than from redesigning the rendering engine.

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 425-446
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

V. CONCLUSION AND RECOMMENDATION

This study examined progressive PDF slice rendering as a user-facing latency problem rather than as a purely graphical optimization problem. Using reconstructed FUNSD forms, measured clip-render traces, and cross-validated tile-utility prediction, the paper showed on this controlled subset benchmark that layout-aware AI can materially reduce time-to-functional-first-frame. The proposed AI-RF scheduler reached TTFF-90 in 14.21 ms on the main 8×10 setting, outperforming the best non-AI heuristic, full-page rendering, and row-major visible-first scheduling by meaningful margins. The result establishes that semantic ordering, not just total raster cost, determines whether a renderer feels responsive during the earliest stages of page exposure.

The paper also identified the conditions under which progressive rendering is advisable. First, the viewer should rank visible slices by predicted utility rather than by simple geometry. Second, the viewer should prefer a moderate grid, with 6×8 performing best in the present evaluation. Third, the viewer should monitor density and fall back to a full-page render when the visible region is saturated with content and clip overhead overwhelms semantic variance. These three rules form a concrete deployment recipe for PdfRenderer-style systems that need lower TTFF without the runtime cost of heavyweight multimodal models.

Several limitations qualify the conclusions. First, the benchmark covers 26 reconstructed documents from the FUNSD test split rather than all 50 test documents. Second, the pages were reconstructed as vector PDFs from annotations instead of using the original scanned raster pages, which isolates layout effects but may not capture all scanner artifacts and background complexity. Third, all timings were collected on one rendering stack and one hardware environment, so absolute latencies should be expected to shift elsewhere even if the relative ordering trends remain similar. These limitations do not negate the measured differences, but they bound the strength of the generalization claims and motivate replication on native PDFs, larger corpora, and additional deployment targets.

The main recommendation is therefore straightforward. A production renderer that supports clipped page jobs should enable layout-aware progressive scheduling by default, use a coarse-to-moderate tile grid, and retain an automatic full-page fallback for dense viewports. Under those conditions, semantic slice prioritization delivers a real reduction in functional latency and produces a page that becomes readable earlier, which is exactly the user-visible benefit TTFF optimization is supposed to achieve.

Author CrediT

H.W. contributed to the conceptualization, methodology development, experimental design, implementation, data analysis, and manuscript preparation. Y.R. contributed to the

conceptualization, supervision, validation, interpretation of results, manuscript review, and project coordination. X.C. contributed to data curation, visualization, literature review, and manuscript revision. All authors have read and approved the final manuscript. H.W. and Y.R. contributed equally to this work and shared first authorship. Y.R. served as the corresponding author.

REFERENCES

Achanta, R., Hemami, S., Estrada, F., & Susstrunk, S. (2009). Frequency-Tuned Salient Region Detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 1597–1604. https://doi.org/10.1109/cvpr.2009.5206596

Appalaraju, S., Jasani, B., Kota, B. U., Xie, Y., & Manmatha, R. (2021). DocFormer: End-to-End Transformer for Document Understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 993–1003. https://doi.org/10.1109/iccv48922.2021.00103

Artifex Software. (2024). *MuPDF Documentation*. https://mupdf.com/docs

Binghua Zhou, Siming Zhao, & David Chao. (2023). LLM-Guided Energy-Aware A/B Testing for Consolidation and DVFS Policies via Power-Sensitivity Clustering. *Journal of Advanced Computing Systems*, *3*(4), 12–30. https://doi.org/10.69987/jacs.2023.30402

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/a:1010933404324

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates. https://doi.org/10.1201/9780203736456

Daren Zheng, & Chenyu Li. (2024). Behavior-Level Jailbreak Resistance via Multi-Stage Refusal + Utility Preservation. *Journal of Advanced Computing Systems*, *4*(1), 83–99. https://doi.org/10.69987/jacs.2024.40107

Fedhira, & Prianto, C. (2025). Systematic Literature Review: Analysis of AI Implementation for Document Verification. *Jurnal Ilmiah Sistem Informasi*, *4*(3), 417–430. https://doi.org/10.51903/kjjwk708

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Harley, A. W., Ufkes, A., & Derpanis, K. G. (2015). Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 991–995. https://doi.org/10.1109/icdar.2015.7333910

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. https://doi.org/10.1109/cvpr.2016.90

Huang, Y., Lv, T., Cui, L., Lu, Y., & Wei, F. (2022). LayoutLMv3: Pre-Training for Document AI with Unified Text and Image Masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, 4083–4091. https://doi.org/10.1145/3503161.3548112

International Organization for Standardization. (2008). *ISO 32000-1:2008 Document Management - Portable Document Format - Part 1: PDF 1.7*. https://www.iso.org/standard/51502.html

Itti, L., Koch, C., & Niebur, E. (1998). A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(11), 1254–1259. https://doi.org/10.1109/34.730558

Jarvelin, K., & Kekalainen, J. (2002). Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, *20*(4), 422–446. https://doi.org/10.1145/582415.582418

Jaume, G., Ekenel, H. K., & Thiran, J.-P. (2019). FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, *2*, 1–6. https://doi.org/10.1109/icdarw.2019.10029

Katti, A. R., Reisswig, C., Guder, C., Brarda, S., Bickel, S., Hohne, J., & Faddoul, J. B. (2018). Chargrid: Towards Understanding 2D Documents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4459–4469. https://doi.org/10.18653/v1/d18-1476

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, *2*, 1137–1143. https://dl.acm.org/doi/10.5555/1643031.1643047

Miller, R. B. (1968). Response Time in Man-Computer Conversational Transactions. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, 267–277. https://doi.org/10.1145/1476589.1476628

Mozilla and individual contributors. (2024). *PDF.js Documentation*. https://mozilla.github.io/pdf.js/

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann. https://doi.org/10.1016/b978-0-08-052029-2.50007-3

O'Gorman, L. (1993). The Document Spectrum for Page Layout Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(11), 1162–1173. https://doi.org/10.1109/34.244677

Oktavia, A., & Wibowo, A. (2025). A New Theoretical Framework for Analyzing the Social and Economic Impacts of AI Within the Digital Economy. *Journal of Management and Informatics*, *4*(2), 859–871. https://doi.org/10.51903/jmi.v4i2.156

Orinos, N., Onola, Q., & Chistoff, O. B. (2025). Zero-Shot Learning for Multilingual Document Classification in Low-Resource Languages. *Journal of Technology Informatics and Engineering*, *4*(3), 391–402. https://doi.org/10.51903/jtie.v4i3.446

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems 30*, 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-abstract.html

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, *1*(6), 80–83. https://doi.org/10.2307/3001968

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). LayoutLM: Pre-Training of Text and Layout for Document Image Understanding. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1192–1200. https://doi.org/10.1145/3394486.3403172

Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florencio, D., Zhang, C., Che, W., Zhang, M., & Zhou, L. (2021). LayoutLMv2: Multi-Modal Pre-Training for Visually-Rich Document Understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2579–2591. https://doi.org/10.18653/v1/2021.acl-long.201

Yifei Lu, Jinyi Mu, & Thao Tran. (2024). Uncertainty-Aware Uplift Modeling for Safer Marketing Targeting: Conformal Prediction and Bayesian Calibration with LCB Policies. *Journal of Advanced Computing Systems*, *4*(5), 84–101. https://doi.org/10.69987/jacs.2024.40507

Yunhe Li. (2024). Findable then Explainable: Retrieval–Summary Integration for Code Intelligence on a Lightweight CodeSearchNet Subset. *Journal of Advanced Computing Systems*, *4*(7), 65–82. https://doi.org/10.69987/jacs.2024.40706

Zhong, X., Tang, J., & Jimeno Yepes, A. (2019). PubLayNet: Largest Dataset Ever for Document Layout Analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 1015–1022. https://doi.org/10.1109/icdar.2019.00166