A higher layout score does not automatically mean a better document pipeline. We learned that while testing Surya OCR 2 against a Paddle layout detector on public-safe historical documents.

Surya reached a higher aggregate Region F1 on two labelled samples. It also found more text regions. But it produced fragmented boxes, weaker label agreement on one sample, and incomplete non-text coverage. When we used its boxes to drive downstream OCR, the extra crops often duplicated or split text that was already covered.

The result was useful, but narrower than "Surya beats Paddle." It showed why OCR layout evaluation needs several metrics and a visual audit before a detector becomes the default.

The short version

- Surya OCR 2 completed a 60-page layout run with no failures or warnings.
- On that sample, Surya's Region F1 was 0.669536, versus 0.619309 for Paddle.
- The gain came mainly from stronger text-region recall.
- Surya emitted no figure or separator regions in that scored view, while Paddle recovered some figures.
- A second 15-page dataset again favored Surya on Region F1, but lower precision and over-segmentation persisted.
- Surya-selected crops could be read by FireRed, but many additional crops were fragments or duplicates rather than new document evidence.
- We kept Paddle as the default detector and Surya as a diagnostic challenger.

What Surya OCR 2 actually does

Surya 2 is a compact document model that supports OCR, layout analysis, reading order, and table recognition. The upstream release describes a roughly 650M parameter model served through vLLM on NVIDIA GPUs or llama.cpp on CPU and Apple Silicon. The public model card reports about 0.7B parameters.

Those upstream claims establish the intended capabilities. They do not establish how Surya behaves on our documents. Our findings below are local reproductions on an RTX 3090 Ti and public-safe labelled samples.

Sources: Surya repository, Surya OCR 2 model card, and Surya 2 release notes.

The layout question we tested

The practical question was not "Which model has the highest score?" It was:

> Can Surya replace Paddle as the detector that finds page regions before OCR?

That role requires more than finding text. A useful detector must preserve the regions, labels, and order needed by the downstream workflow without creating excessive duplicate or fragment crops.

We separated four things that are often mixed together:

Evidence type	What it measures	What it does not measure
Layout Region F1	overlap between predicted and labelled regions	transcription accuracy
Label accuracy	whether matched regions have compatible labels	whether all regions were found
Reading-order agreement	order among matched regions	order for missing or extra regions
Crop-text CER and WER	OCR text after a detector selected crops	native full-page Surya OCR quality

This separation matters. A layout-only run does not execute Surya transcription. A Surya-plus-FireRed crop experiment measures FireRed text inside Surya-selected boxes, not Surya OCR text.

Test 1: 60 public-safe historical pages

The first layout comparison used 60 pages from the public-domain IMPACT KB collection with PAGE XML region annotations. Both model outputs were scored at an intersection-over-union threshold of 0.5.

Surya processed all 60 pages:

Metric	Surya OCR 2 layout	Paddle layout
Region recall	0.594269	0.533477

Region precision	0.795282	0.788249
Region F1	0.669536	0.619309
Label accuracy on matches	0.799619	0.855772
Reading-order agreement	0.960977	0.987638
Predicted regions	555	510

Surya had the higher F1 on 35 pages. Paddle led on 13, and 12 tied. The median Surya-minus-Paddle F1 difference was 0.041452.

At first glance, that looks like a clean Surya win. The label-level results changed the interpretation.

The aggregate score hid a coverage gap

On the 60-page sample, the PAGE XML truth contained 429 text regions, 278 figure regions, and 8 separators.

Surya predicted:

- 509 text regions
- 0 figure regions
- 0 separator regions

Paddle predicted:

- 438 text regions
- 9 figure regions
- 0 separator regions

Surya's higher aggregate F1 came from finding more text. That can be valuable for text extraction, but it is not broad layout superiority. A workflow that needs figures, separators, or stable semantic labels would read the result differently from a workflow that only wants text blocks.

Visual review found fragments and duplicates

We manually audited 18 pages and recorded 163 issues:

Issue	Count
Missed truth regions	79
Duplicate or fragment boxes	60
Wrong layout labels	23

Under-segmentation	1

Many Surya boxes covered real visible text. The problem was granularity: a box sometimes represented a fragment inside a larger labelled region. Such a box may be a useful OCR crop, but it is not a clean one-to-one replacement for the PAGE XML region.

Test 2: a second labelled dataset

We then used 15 pages from OCR-D as an independent public-safe layout sample.

Metric	Surya OCR 2 layout	Paddle layout
Region recall	0.648677	0.359630
Region precision	0.740605	0.876667
Region F1	0.648779	0.492342

Surya again led on Region F1, this time through substantially higher recall. It also matched all 4/4 labelled figure regions in this second sample.

That supports Surya as a serious detector candidate. It does not erase the first sample's missing figure coverage or the repeated over-segmentation pattern. The defensible conclusion is dataset-sensitive: Surya often finds more regions, while Paddle can remain more conservative and precise.

What happened when Surya boxes drove OCR

We ran two 15-page diagnostic pipelines in which Surya selected text-like regions and FireRed transcribed the resulting crops.

Sample	Surya crops	Strict matched	Strict unmatched	Crop failures
Extra 15 pages	104	73	31	0
Original 15 pages	142	98	44	0

Operationally, the pipeline worked: FireRed returned text for every requested crop. Interpretively, the additional boxes were often not new information.

Across the two samples, fallback review found:

- 48 contained fragments
- 15 duplicate-truth matches
- only 1 new-truth match

This is the central lesson. More boxes can mean better recall, but they can also mean repeated OCR, fragmented text, inflated crop counts, and harder reconciliation.

Full-page OCR had a different failure mode

We also ran Surya full-page OCR on 60 public-safe pages. All 60 completed, with median latency of 11.671 seconds per page. Two pages triggered loop fallback and took roughly two minutes each.

Condition	Median CER	Median WER	Median latency
Non-fallback pages	0.052844	0.191549	11.593 s/page
Fallback pages	0.227454	0.274217	124.946 s/page

One fallback page also contained suspicious Cyrillic and Armenian characters in noisy regions. These two pages should not be blended into ordinary Surya output without an explicit fallback flag.

This is separate from the layout result. A detector can be promising while the full-page generation path still needs fallback-aware reporting and latency controls.

Why Surya did not replace Paddle

We kept Paddle as the default detector because the replacement decision required all of the following:

- stable region coverage across document families
- acceptable label coverage, not only text recall
- reliable reading order
- manageable crop counts
- explicit duplicate and fragment handling
- predictable fallback and latency behavior
- a stable adapter and reporting contract

Surya demonstrated several of these, but not the complete package. Its higher F1 was real within the tested samples. The inference that it should replace Paddle was not.

A better way to evaluate OCR layout

When comparing layout detectors, use this sequence:

1. Score recall, precision, and F1 separately.
2. Break results down by label, especially text, table, figure, heading, and separator.
3. Report reading order only for matched regions and disclose that boundary.
4. Count extra boxes, fragments, duplicates, and missed regions.
5. Inspect the pages with the largest model disagreement.
6. Test whether downstream OCR benefits from the extra crops.
7. Keep detector scores separate from transcription scores.
8. Require a second document family before changing the default.

This approach turns a leaderboard number into an operational decision.

When Surya OCR 2 is worth testing

Surya remains a good candidate when:

- missing text regions is more expensive than processing extra boxes
- multilingual document support matters
- you need one toolkit spanning OCR, layout, order, and tables
- you can preserve fallback status and audit fragmented regions
- you are willing to tune the detector-to-OCR boundary for your corpus

Keep a conservative detector in the comparison when figures, separators, stable semantic labels, or predictable crop counts are important.

Licensing boundary

The Surya repository and model weights do not share a simple permissive license. The repository identifies GPL code, while the current model weights use an OpenRAIL-style license with use conditions. Review the exact code revision, checkpoint license, organization eligibility, deployment model, and redistribution

plan before commercial adoption. Do not infer commercial permission from model availability alone.

Final verdict

Surya OCR 2 was not rejected. It was kept as a diagnostic challenger.

It achieved higher layout F1 than Paddle on both public-safe samples we tested and showed strong text-region recall. But aggregate F1 hid label-coverage differences, fragmented boxes, duplicate crops, and a more complicated downstream contract. Full-page OCR added a separate fallback and latency-tail concern.

The useful question is not "Did Surya win the benchmark?" It is "Did Surya make the complete document pipeline more reliable?" Our current evidence does not yet justify that stronger claim.

For the broader model-routing decision, read Which OCR Model Fits Which Workflow in 2026. For our benchmark design, see How We Benchmark OCR Models on Scan-Heavy PDFs.