Using an Ensemble Approach for Layout Detection and Extraction from Historical Newspapers

Aditya Jadhav
Department of Computer Science
Virginia Tech
Blacksburg, USA
ORCID: 0009-0005-0507-3721

Bipasha Banerjee
University Libraries
Virginia Tech
Blacksburg, USA
ORCID: 0000-0003-4472-1902

Jennifer Goyne
University Libraries
Virginia Tech
Blacksburg, USA
ORCID: 0009-0006-0037-5322

***Abstract*—Digitized newspapers are archival resources that contain complex and inconsistent layouts. Pages often include multiple columns, large headlines, images, and advertisements, along with scanning issues such as noise or uneven lighting. Cloud services can provide a quick baseline, yet per-page pricing and proprietary models limit full-collection processing, and transparent evaluation. We present an open-source workflow that approaches layout understanding as an essential step for newspaper digitization. The pipeline runs in iterative stages: it first applies OpenCV rules to extract high-confidence text panels, then uses the Newspaper Navigator model to remove images and advertisements, and finally applies a fine-tuned Detectron2 model to recover remaining text regions. Each detection is saved as a crop with its coordinates and metadata recorded in a manifest file. Text crops are transcribed using a vision-based multi-modal large language model (LLM). This open source service is scalable, supports consistent reprocessing, and produces useful outputs that support archival goals of access and preservation.**

***Index Terms*—Document Layout Analysis, Historical Newspapers, Text Region Detection, Optical Character Recognition, Vision Large Language Models**

I. Introduction

Making digitized historical newspapers machine-readable at scale is problematic. Some of these challenges include complex layouts with graphics, multi-column text, dense textual elements wrapped around images, and poor-quality scans resulting in fading, bleeding, guttering, and skewing. Out of the box, existing methods typically expect pages as a single object containing homogeneous content, which results in errors when applied to digitized newspapers. These challenges require approaches that work with complex layouts to preserve the content's reading order.

Existing approaches to make newspapers more machine-readable fall into three board classes, end-to-end cloud services providing a baseline, page costs limit multiple processing, and fee based service models offer limited flexibility. When the model confuses photos made of dot patterns (halftones) for text, merges warped gutters, or breaks complex headlines, we cannot adjust or tune the thresholds, and the results can vary across runs. These constraints complicate the archival needs for provenance, reproducibility, and targeted remediation. Second, single deep detectors on full pages face domain shift. On our material, they failed to detect regions. In some cases, they produced small, unstable boxes that needed post-processing removal. Cluttered input forces one model to learn everything at once, which hurts calibration. Third, rule-based pipelines are fast and precise on clean columns, but break under skew, bleed-through, advertisement clusters, and complex borders. In short, no single method works on archival newspaper pages.

Our approach focuses on making the detection task easier by simplifying the page before applying deep learning models to interpret it. We separate the layout detection phase from the text detection phase and process the page in several passes. The pipeline runs in three stages: an OpenCV [1] layer first identifies clear text panels; the Newspaper Navigator [2] then removes images, advertisements, and cartoons; and a fine-tuned Detectron2 [3] model finally recovers any remaining text regions. Each pass refines the page further until no new text is found. This layout-first design follows prior findings that recognizing structure early improves both optical character recognition (OCR) accuracy and reading-order quality in newspaper collections.

This design directly addresses several common problems in previous systems. Starting with simple geometric rules helps prevent early mistakes such as merging separate columns or erasing faint text. By removing non-text regions before text detection, we reduce false positives at advertisement borders and near halftones, while making detection more stable on headlines, captions, and narrow columns. Running a text-only detector on a cleaned page reduces the class confusion between text and graphics, improves calibration, and recovers small or hard-to-read text areas. All detections are refined based on their geometry alongside model confidence. After detection, we apply several cleanup steps to make sure the boxes are logical. We remove duplicates and nested boxes, resolve overlaps between classes based on priority rules, and filter out boxes that are clearly too large or too small. These checks ensure that even faint or small text is kept, while reducing noise from incorrect detections.

Equally important for archival practice, the workflow is designed to be fully repeatable. Every detected region is saved as an individual crop with its coordinates and metadata recorded in a page-level manifest. Intermediate results, including thresholds, overlays, and decision logs, are also stored so that any error can be traced to the exact step that produced it. Text extraction is performed on each segment with a vision-

based model, and the reading order is reconstructed from geometry rather than model labels. This approach is applied to keep the order consistent across domains and preserve the original visual layout of the page.

Our research contributions are as follows:

- A pipeline that separates layout detection from text extraction and uses iterative processing to improve stability on historical pages.
- A multi-layer layout recovery process: precision-first OpenCV detections, Newspaper Navigator for non-text removal, and a fine-tuned Detectron2 text-only detector on the cleaned page, with geometry-based reconciliation.
- A dedicated text-only detector fine-tuned on the PRImA Layout Analysis Dataset [4], referred to as *TextOnly-PRIMA* to avoid loose, overlapping, and multi-class boxes produced by generic document detectors.
- Segment-level transcriptions that preserve page logic, paired with manifest files that record each step and supports clear review of extracted content.

II. Literature Review

Document layout analysis has become a central component of document image understanding. It combines computer vision and information extraction to detect structural components such as text blocks, headlines, figures, and tables that support downstream OCR tasks. Recent surveys describe layout analysis as the foundation of modern document parsing, emphasizing its role in converting visual content into structured digital representations [5]. The diversity of page layouts in digitized historical newspapers prevents a single method from working consistently, as described in [6]. Models trained on comparable set of documents often lose accuracy when applied to unseen or structurally different material. Newspaper pages combine multi-column text, advertisements, use specific ink for newsprint, making reliable segmentation a prerequisite for recognition. To address this variability, Zhu et al. [7] introduced DocBed and the NewsNet7 dataset of annotated newspaper pages, demonstrating that segmentation accuracy directly influences downstream OCR quality. Fleischhacker et al. [8] reached similar conclusions on the nineteenth-century printed comprehensive civil records of Hasburg titled *Hof- und Staatsschematismus*. The authors fine-tuned a Faster R-CNN and showed that accurate region detection reduced character and word error rates by more than fifteen percentage points. Collectively, these studies confirm that precise, layout-aware structure detection is the foundation on which reliable OCR in historical collections depends.

Commercial OCR systems integrate optical character recognition with layout parsing and large-scale orchestration to provide end-to-end text extraction services. These platforms, such as AWS Textract [9], Tesseract [10], and Google Document AI [11], are designed for ease of deployment through cloud interfaces that automate document ingestion and output processing. Bawasker et al. [12] describe the AWS Textract workflow, highlighting its interoperability with other Amazon Web Services, including S3 and Lambda, and its capacity to extract tables and key–value pairs from structured forms. The study points out Textract's engineering convenience and scalability within the AWS ecosystem. As with most commercial cloud OCR services, the underlying models are proprietary and subscription-based. This model can be a constraint for research universities operating on a limited budget, and transparency for research replication.

Layout detection methods have progressed from rule-based segmentation toward deep and semi-supervised learning. This shift reflects the broader evolution of document image analysis from handcrafted heuristics to data-driven predictions. Early systems used geometric and morphological operations to separate visual structures such as tables and columns. The authors in [13] demonstrated that OpenCV-based line detection can reliably identify table boundaries while maintaining computational efficiency, offering a lightweight option for routine layout tasks. With the shift to deep learning, convolutional neural networks and transformer architectures now dominate document segmentation. Grüning et al. [14] introduced the ARU-Net, a baseline detection network that labels pixels as text lines or separators and clusters them into coherent lines. The method showed strong accuracy on benchmark datasets with limited training data. A subsequent study in [15] applied a YOLOv3 [16] detector with a feature pyramid network to handwritten historical pages, improving sensitivity to small and irregular regions. Research on printed layouts in [7] benchmarked fully convolutional, transformer, and hybrid segmentation models on the NewsNet7 dataset. The authors found that the transformer-based segmentation network achieved the highest performance on newspaper pages. In low-resource settings, the evaluation in [17] found YOLOv8 [18] to perform best among current detectors for Bengali documents. Collectively, these developments illustrate a steady progression from rule-based systems to adaptive neural architectures that improve accuracy and reduce dependency on large annotated datasets.

Open tools and reproducibility frameworks have improved the transparency of document layout research. Shen et al. [19] introduced LayoutParser, an open-source toolkit that unifies layout detection, visualization, and integration with OCR engines within a consistent API. Alberti et al. [20] developed an open evaluation suite that reports precision, recall, F1, and intersection-over-union within a standardized evaluation framework. These frameworks reduce implementation differences and enable reproducible, benchmark-driven comparisons across studies.

Optical character recognition models have advanced with the introduction of multi-modal vision language models that can directly perform OCR on historical documents. Researchers in [21] evaluated several open-source technologies, including vision LLMs, and concluded that *Mistral* [22] and *olmOCR* [23] achieved the best performance among the models tested.

Evaluation metrics commonly used for document layout detection are average precision (AP) and mean average precision (mAP). These metrics follow the COCO-style [24] evalua-

tion protocol, which standardizes measures across intersection-over-union thresholds. AP represents the area under the precision–recall curve for a single class, while mAP averages this value across all classes to provide an overall measure of detection accuracy. Results are often reported at intersection-over-union (IoU) thresholds such as AP@50 or AP@75, where higher thresholds indicate stricter agreement between predictions and ground truth. These metrics are widely used to compare model performance in document layout analysis.

III. Dataset

Our experiments use digitized newspapers made available through the Montgomery Museum (Montgomery County, Virginia) [25]. The museum is a nonprofit archive that collects and preserves regional history and maintains a library of books, documents, genealogical records, and photographs for public use. As of the most recent snapshot we worked from (updated October 2024), the collection provides instant access to several local titles, including the Montgomery Messenger, Montgomery News, Montgomery News Messenger, Radford Enterprise, Radford News, Radford News Journal, Radford Record and Advance, and The News Journal. More information about this collection is available online [25]. These newspapers span multiple decades of community reporting and exhibit the full range of historic newspaper layout features, dense multi-column text, varying gutter widths, ornate headlines and drop caps, and frequent inclusion of advertisements, photographs, and tables.

Montgomery newspaper images originate from mixed digitization workflows (flatbed scans and camera captures), so resolution, contrast, and skew can vary. This variability is representative of a newspaper collection and is useful for testing layout methods. We have encountered common historical artifacts such as page bleed-through, local warping, uneven illumination, and occasional issues along the seam of the paper.

IV. Methodology

A. Overview

This section introduces the open-source pipeline for structural analysis and text extraction in archival newspapers. It was developed following a controlled AWS Textract baseline and redesigned for flexibility, transparency, and large-scale use. We first discuss the AWS baseline approach that motivates the open approach, and finally describe an iterative pipeline that separates layout detection from text extraction.

The workflow operates in three main stages that repeat until no new regions remain. Each pass begins with a conservative OpenCV stage that captures clear text panels. This is followed by a detector that removes non-text content such as images and advertisements, and finally a text-only model that recovers remaining regions. After every stage, accepted detections are cropped, saved, and logged to a page-level manifest file. The process continues for a small, fixed number of rounds or until the unexplained area becomes negligible. The output is a page reconstructed as a set of segments, each with its coordinates, label, and source image reference that can be used for article assembly, OCR, and archival description. All stages are fully reproducible, and every decision is stored for later review.

We begin by describing the AWS Textract baseline, which established our reference architecture and motivated the transition toward an open, reproducible workflow.

B. AWS Baseline

We begin with an end-to-end implementation using AWS Textract to establish a working reference system. A user upload to Amazon Simple Storage Service (S3) triggers a sequence of serverless steps, as illustrated in Fig. 1. Each phase ran as an independent Lambda function connected through AWS SNS for event notifications.

Layout Detection. The first Lambda function submits a layout-detection job to Textract and publishes a notification to Amazon SNS. Upon completion, a second Lambda function saved the layout artifacts back to S3 so that bounding boxes and detected regions could be inspected quickly. This layout detection step is part of the workflow, in which the system first processes incoming data to extract page layout features.

Text Extraction. A third Lambda function initiates a text-extraction job on the layout-detected pages. Once the recognition results arrive through SNS, a fourth Lambda function stores the raw JSON response. This phase, which involves text extraction from the detected regions, is completed after the layout detection.

Post-Processing and Parsing. In the final stage, another Lambda function parses the stored JSON response into our internal manifest. This manifest records block identifiers, hierarchical PAGE-REGION-LINE-WORD links, normalizes coordinates in page space, and basic quality flags using the `LINE`/`WORD` structures exposed by Textract's API [9]. The parsed information is stored and organized for later processing.

This architecture supports consistent batch processing, and thus removes the need for manual supervision. With the pipeline running, we used AWS Textract LINE and WORD boxes to prototype a geometry-based reading order that groups text by vertical bands, identified columns through horizontal overlap, and sorted lines within each column.

The AWS baseline showed several strengths. It rarely returned empty pages even under noise, confidence scores proved useful for triage and human review, and the asynchronous job model scaled well for batches. The combination of S3, SNS, and Lambda made orchestration simple and supported rapid iteration across multiple titles, consistent with prior reports on Textract's ease of orchestration in cloud workflows [12]. These properties enabled us to build intuition about both common and unusual newspaper layouts and to collect examples of success and failure for subsequent experiments.

However, several limitations became apparent as we planned full collection runs. First, the cost at scale is not sustainable due to the volume of pages. Second, limited control over the detection threshold proved to be problematic for historical material. Third, small variations across runs complicated reproducibility. These factors led to an open detector-agnostic

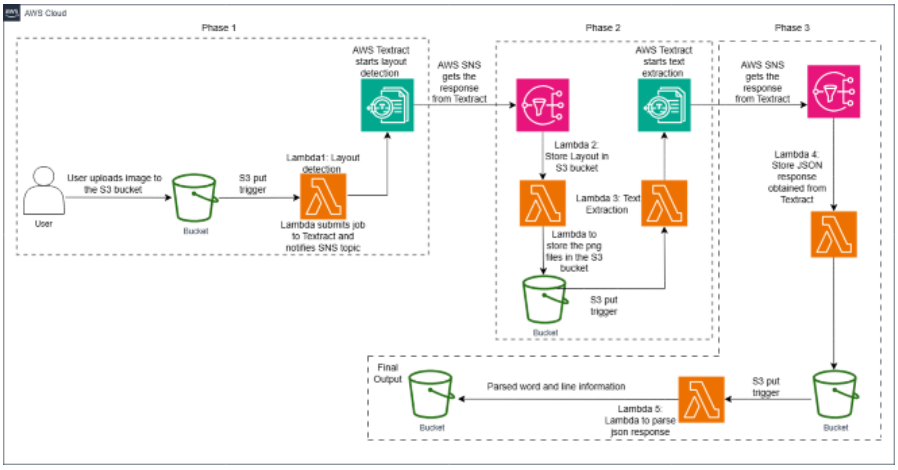


Fig. 1: *AWS Textract baseline architecture: The pipeline is triggered by Simple Storage Service (S3) uploads via Lambda service and responses sent via Simple Notification Service (SNS).*

architecture that retains operational strengths, batch automation, and scalability by reducing cost and control barriers for large-scale processing.

C. *Open-source Methodology: Rationale for the Pivot*

The AWS baseline proved two practical constraints at once: structural analysis is the limiting factor for historical newspapers, and cloud pricing becomes the constraint as soon as we move from a pilot to a collection at scale. The library hosts an array of collections, some as large as 8 TB, with thousands of items within each of these collections. Therefore, only using proprietary models for processing large collections is not cost-effective. In addition to initial runs, we need to re-run the process when models improve. Per-page fees that are acceptable for feasibility testing become a hard blocker at that scale. At the same time, managed service and proprietary models offer limited control over thresholds and class priors, which makes it difficult to tune for the realities of historic materials. We therefore moved to an open pipeline that can be re-run at low marginal cost and tuned directly for our collection, producing outputs that can be easily evaluated.

Early experiments with off-the-shelf open models confirmed that no single method can handle the full range of newspaper pages on its own. A deep detector run on the full page often missed large regions or split regular columns into many small, low-confidence boxes because the model attempted to explain text, halftones, and borders at once. Purely rule-based approaches worked well on clean, rectangular columns, but failed when skew, show-through, drop caps, or mixed graphics appeared. These observations led us to an ensemble pipeline that reduces visual complexity before applying any deep learning model. We begin with conservative OpenCV operations that detect only the most regular blocks with high precision. We then apply a newspaper-specific detector to remove large non-text elements such as photographs, advertisements, and cartoons. Finally, we run a fine-tuned text-only detector model that identifies only the text regions. We resolve overlaps using geometry between boxes rather than relying on the detector's confidence scoring. We end the process when additional passes do not produce new regions.

To make the approach more resilient, we craft a set of non-negotiable design requirements such as low marginal cost for reprocessing, fine-grained tunability, stability before recall, geometry-aware reconciliation, and downstream continuity.

D. *Phase 1: Layout Detection*

The newspaper pages are run as an iterative cycle, isolating the easiest structural elements, then reducing image clutter on the cleaned images and finally applying learned detectors. The loop is intentionally simple and fully reproducible. We start with the OpenCV-based operations to capture high-confidence block regions-panels framed by strong boundaries or dark edges. These regions are cropped and stored with page-level coordinates, then their pixels are masked to create a cleaned page that contains only unexplained regions. The cleaned page is then passed to a Newspaper Navigator model [19], which identifies photographs, advertisements, and cartoons, which are cropped, labeled, and masked. Finally, the remaining areas are analyzed with `TextOnly-PRIMA` to recover additional text regions that become easier to detect as the image becomes more uniform. Each iteration produces new crops, a cleaner page, and a detailed record of all changes. Each iteration further simplifies the page, allowing later detectors make fewer errors and reconciliation becomes easier.

1) ***Layer 1: OpenCV****:* Each page is first processed using OpenCV because it identifies the best layout components with very hight precision. OpenCV can identify framed blocks, such as advertisements or column panels, based on contour and boundary cues. The objective at this stage is to find stable well-bounded regions while avoiding changes that might hinder detection. The process begins by locating the physical page using Otsu thresholding method. This processing method automatically detects black-and-white cutoff based on the image's brightness distribution. Next, the mask is inverted, darker printed areas appear white on a black background, and the largest contour is selected to represent the page boundary.

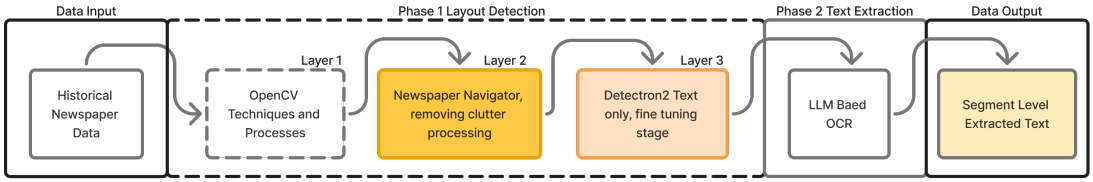


Fig. 2: *Iterative pipeline for historical newspaper analysis. Phase 1 performs residual-refinement with OpenCV, Newspaper Navigator, and a text-only detector. Phase 2 performs OCR extraction.*

We slightly erode this contour, shrinking it by a few pixels to ignore artifacts near the scanner edge. Inside this page mask, we convert the page to a binary image using adaptive thresholding to remain robust under uneven lighting.

We then apply light morphological operations, such as small-scale opening and closing, to merge nearby strokes or dark lines into solid blocks while keeping column gaps intact. From this processed image, we extract connected components, groups of touching pixels and filter them by size, shape, and solidity to focus on dense rectangular regions bounded by clear edges rather than scattered noise. When a region meets these conditions, it is cropped and saved with both the crop and page coordinates, and the corresponding pixels are masked to divide the page into explained and unexplained areas, so later stages can focus on what remains unresolved.

To make the detection more stable across different page styles, we perform two complementary preprocessing passes.

1) **Big-box pass:** In the first pass,we enhance contrast using CLAHE [26], which improves local contrast without over-amplifying noise. We also remove slow background variations using a large morphological open and then apply a broad closing operation after thresholding to fill in small gaps. This step highlights large framed areas such as half- or full-page advertisements or clearly bordered columns.
2) **Standard pass:** In the second pass, we generate a line map using vertical and horizontal black-hat filters, which highlight dark lines on light backgrounds to emphasize gutters and text lines. We also created an ink map by applying adaptive Gaussian thresholding, followed by small opening and closing operations to connect nearby strokes and suppress speckles.

Candidate boxes derived from these maps are filtered by simple geometry. Boxes that are too small, fall outside a safe area band, or touch a narrow edge margin are discarded. This keeps the stage focused on strong, well-defined blocks and prevents outliers from influencing later scoring. For each remaining candidate, we compute a compact set of features to distinguish likely layout regions from noise. The features include, ink fraction, texture strength measured by Laplacian variance, canny edge density, a count of small connected components in a normalized band, regularity of text line spacing estimated using a Fourier transform of the horizontal projection profile, border contrast (difference in edge energy between the crop border and a thin outer ring), and line coverage from the line map.

Conservative filters are applied before scoring. Long, thin seams are removed when the aspect ratio is high, but both ink fraction and periodicity are low. Low-texture patches are removed when Laplacian variance is below a threshold. Regions with edge density or ink fraction outside expected ranges are eliminated, and candidates lacking periodic text lines, border contrast, or line coverage are also removed. Each remaining box is assigned a scalar score that prioritizes strong border contrast, line coverage, periodicity, edge density, Laplacian variance, and small component count while penalizing flat or seam-like regions.

Overlapping detections are resolved through non-maximum merging and nested-box removal, keeping the larger region whenever one lies fully inside another. The remaining boxes are then aligned to nearby strong lines within a small pixel tolerance so that edges match visible separators and are padded slightly to avoid clipping. Finally, we retain only the top K regions per page (with $K = 2$ in our precision-first configuration). Several practical lessons shaped these choices. Smaller kernels preserved line spacing and maintained gutter cues that were useful for reconstructing reading order. Solidity and convexity checks helped suppress halftone fragments and decorative frames that might otherwise appear as valid panels. Larger kernels improved recall but tended to merge adjacent columns, so a more conservative configuration was preferred to protect faint text strokes and marginal details that hold archival value.

The first cleaned page image is then produced by masking or lightly filling in the accepted block pixels, thereby directing the next detectors toward the remaining unresolved areas. For transparency and reproducibility, each crop is saved with its coordinates, feature record, and score, along with debug layers such as the page mask, normalized image, line map, and ink map. The result is a small, high-confidence set of block regions and a clean separation between processed and unprocessed areas, preparing the page for the following *Newspaper Navigator* and *TextOnly-PRIMA* stages.

2) ***Layer 2: Newspaper Navigator****:* After the OpenCV stage removes well-bounded block regions and we create the first cleaned page, the page still contains the visually dominant non-text elements typical of historical newspapers. These include photographs, advertisements, logos, and editorial cartoons that occupy large rectangular areas and often exhibit strong edges or halftone patterns. We process this residual using `LayoutParser` with the *Newspaper Navigator* detector, which is specifically trained for these visual categories and consistently removes them early in the pipeline.

The intent is straightforward: to eliminate large distractors so that subsequent text detectors do not compete with high-contrast graphics.

We loaded the Newspaper Navigator hub model and fixed two operating points that proved to be stable across our material: a detection score threshold of 0.60 and a non-maximum suppression threshold (NMS) of 0.30. These values maintain a balanced trade-off between precision and recall, without flooding later stages with redundant detections. Because the original label map in the hub contains overlapping or ambiguous keys, we normalize label mappings on load to ensure distinct class identifiers for *Photograph*, *Illustration*, *Map*, *Comics or Cartoon*, *Editorial Cartoon*, *Headline*, and *Advertisement*. This normalization prevents silent label collisions in downstream reconciliation.

During experiments, we observed that the model's *Photograph* predictions were often over sized and would bleed into adjacent text columns. Cropping and subtracting these boxes at this stage, removed nearby text that the next detector needed to identify. To avoid this, we *do not crop or subtract* regions labeled *Photograph* in Layer 2. All other classes (*Illustration*, *Map*, *Comics/Cartoon*, *Editorial Cartoon*, *Headline*, *Advertisement*) are handled normally. Photographs are deferred until after the text-only detection stage (Layer 3), at which point we revisit the page and crop the images more closely if needed. This change preserves text that sits at the boundary of the image and improves downstream recall without losing the ability to capture images later.

Detections are handled identically to OpenCV panels: each detected region is cropped, labeled with class and confidence score, saved with both crop and page coordinates, and recorded in the page-level manifest. Each accepted box is subtracted from the residual to prevent reprocessing in later passes. The impact is immediate; after subtraction, the residual becomes smaller, quieter, and visually closer to the statistics of body text. We also apply a light post-filter to refine Navigator outputs. Extremely thin or tiny detections are dropped; headline candidates spanning nearly the full page width but lacking adjacent text are flagged for review; and clearly nested or duplicate detections are suppressed. These simple checks prevent a single high-scoring but implausible box from erasing content that later stages may still need.

A key lesson from the experiments was that the order of operations matters. When text detection was attempted before removing non-text, the text model produced fragmented boxes, unstable boundaries near advertisement borders, and over-segmented headers adjacent to photographs. Running Navigator first yielded markedly cleaner results, allowing later detectors to process more uniform textures and demonstrate more consistent behavior. This effect was more pronounced in dense advertisement clusters, where halftones and decorative frames often share borders with short captions. By clearing these complex blocks in a dedicated pass, we significantly reduced clutter and improved the downstream signal for text detection.

Some failure cases remain persistent and are addressed with simple geometric rules. The Navigator model occasionally generates oversized *Advertisement* boxes that overlap small caption or photo credits. To mitigate this, acceptance rules include a padding-aware geometric check. For example, if a large non-text region would erase text lines at its borders, the box is trimmed by snapping to the nearest strong separator before subtraction. Conversely, Navigator may miss faint or low-contrast images embedded within dense classified text. These remnants are harmless because the subsequent TextOnly-PRIMA stage focuses on text, and our later reconciliation step can demote such mislabels based on shape sanity checks. After the Newspaper Navigator step, most large non-text regions are removed. The remaining page areas are smaller, more uniform, and visually consistent with body text, providing ideal conditions for the next stage, *TextOnly-PRIMA*. This is specifically designed to detect text regions from within the cleaned residual.

3) ***Layer 3: TextOnly-PRIMA****:* During early experiments with off-the-shelf document detectors such as PubLayNet [19] and Newspaper Navigator [19], we observed several persistent limitations when applied to historical newspaper pages. These generic models, trained on modern digitally typeset documents, frequently produced *loose* text boxes that extended beyond paragraph borders or merged adjacent columns. They also assigned multiple, overlapping label classes to the same physical region (for example, a single header classified simultaneously as *title* and *figure caption*). This multi-class fragmentation inflated the number of bounding boxes, complicated post-processing, and destabilized reading-order reconstruction. As a result, OCR consistency and region-level provenance suffered. These issues made clear that a general-purpose detector was ill-suited for precise segmentation of historical text layouts.

To address these problems, we developed a dedicated *text-only* detector fine-tuned for the historical material. Specifically, we adapted the PubLayNet [19] architecture for single-class detection and retrained it on the *PRImA Layout Analysis Dataset* [4]. The dataset is comprised of 19th to 20th century pages with polygonal ground-truth annotations for text blocks, titles and captions, capturing typical archival challenges such as bleed-through, complex fonts, halftones, and uneven lighting.

The resulting `TextOnly-PRIMA` detector produces tight, uniform bounding boxes around text areas and avoids multi-class overlaps that were common in off-the-shelf models. This simplification reduced downstream complexity, improved OCR alignment, and stabilized reading-order reconstruction. The resulting model, referred to as `TextOnly-PRIMA`, now serves as the third-stage detector in our open pipeline (Fig. 2). It specializes in recovering fine-grained textual regions, captions, narrow columns and headers that earlier stages may miss. Integrating this fine-tuned detector substantially improved page coverage, reduced redundant detections, and stabilized convergence in the iterative residual loop, especially on degraded or visually complex historical pages.

TABLE I: Fine-tuning configuration and results for the TextOnly-PRIMA detector.

Component	Specification / Result
Base architecture	Detectron2 `GeneralizedRCNN` (ResNet-50 FPN)
Training data	`prima_text_train`, `prima_text_dev`; tested on `prima_text_test`
Optimizer	SGD (momentum 0.9, batch 4), LR 1×10^{-4}, decays @ 1000 & 1400 iters
Iterations	1 600 total
Augmentation	Multi-scale (1000–1800 px) + TTA (1200/1400/1600 px + flip)
Anchors	Added 16-px level for small text
Thresholds	Score 0.45, NMS 0.40
Evaluator	COCOEvaluator (PRImA test split)

E. Phase 2: Segment-Level Text Extraction

After the layout detection stage has finished, each detected region is treated as an independent segment for text recognition. In this phase, we perform optical character recognition (OCR) on every region separately and then combine the recognized text to rebuild complete articles. Working at the segment level is important because page-level OCR on historical newspapers often struggles with noisy backgrounds, faded ink, or overlapping advertisements. By focusing only on clean, well-defined regions, the OCR model can read text more accurately and avoid interference from surrounding content. For this step, we use the `allenai/olmOCR-7B-0225-preview` model (OLMOcr-7B), a large multimodal vision–language model designed for high-resolution document reading. Recent work by researchers in [21] evaluated several open-source OCR and vision-LLM systems on historical documents and found that Mistral and OLMOcr achieved the highest accuracy. Based on these findings, we adopted OLMOcr-7B for this project due to its strong OCR performance and open availability. Each region is paired with a short and direct prompt that asks the model to copy the visible text exactly as shown. We use three simple prompt types: one for single-line text, one for multi-line blocks, and one for captions that may appear below figures or images. The model runs with fixed generation settings for consistency, and every region's result is stored together with its bounding box, class label, and model information.

The OCR output from each region is then lightly cleaned to make it consistent. We only apply small adjustments, such as joining words split by hyphens at line breaks and removing extra spaces. After OCR is complete for all regions, the recognized text is grouped into articles using geometric relationships between boxes. Columns are identified by clustering the horizontal midpoints of boxes that are close together. Within each column, two boxes are linked if their vertical gap is small and they overlap horizontally by at least forty percent of their width. All connected boxes form one article, which is then ordered by column and vertical position to match the reading order. Each span keeps its type from the previous phase (headline, paragraph, caption), and figures or tables are kept as placeholders with optional caption text. Every article stores its list of regions and the overall bounding box that covers them.

Running OCR on smaller, well-defined segments gives better accuracy and stability than running it on full pages. The model reads text more reliably, avoids false detections near images or borders, and can process many regions in parallel, which reduces overall runtime. Every run is reproducible, with all settings and files stored for traceability. Overall, this phase converts the detected layout from Phase 1 into clean, structured text that can be easily reviewed, searched, and used in later stages of processing.

V. RESULTS

We analyze the pipeline at each processing layer to understand how the page evolves from raw scan to structured text. Each layer contributes a specific function, starting with high-precision geometric filtering, moving through class-aware layout detection, and ending with segment-level OCR and article reconstruction.

A. AWS Baseline

AWS Textract provided the initial layout-detection baselines. We ran Textract on a small sample of historical newspaper pages. It identified LINE and WORD regions, but the broader region-level segmentation often struggled with dense or irregular column structures. These observations, combined with the cost of running the full collection at scale, led us not to extend Textract beyond the initial evaluation phase.

B. Phase 1: Open Source Layout Detection

*1) **Layer 1: OpenCV-Based Region Detection**:* The first layer applies a contour-based detector that uses adaptive thresholding, connected-component filtering, and basic morphology to locate high-confidence block regions before any pre-trained model is applied. On the page shown in Fig. 3, the detector captures the large, regular panels such as price tables, product headings, and store banners, while effectively ignoring page borders, background noise, and decorative elements. The output is intentionally conservative: the boxes are tight, the gutters remain visible, and only the most regular block areas are extracted. This ensures that subsequent phases receive cleaner inputs.

The OpenCV pass is conservative and fast, working on dense paragraphs and bold framed panels. Common limitations include missing thin caption lines near images, overlooking small decorative labels, or partial merges in mixed text–image advertisement blocks. These trade-offs are acceptable because the goal at this stage is to recover the most reliable text regions and leave complex cases for downstream phases.

Overall, the OpenCV layer serves as an initial filter that extracts the bulk of easily detectable text. By design, it favors precision over recall, removing regular blocks while keeping complex or uncertain regions untouched. The remaining content, mainly advertisements, photographs, and decorative layouts, forms a cleaner and more structured page for the next stage. This page is then passed to the Newspaper Navigator model, which focuses on detecting large non-text elements.

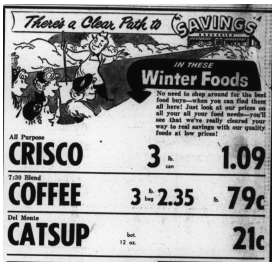


Fig. 3: *Output from the OpenCV-based region detection layer.*

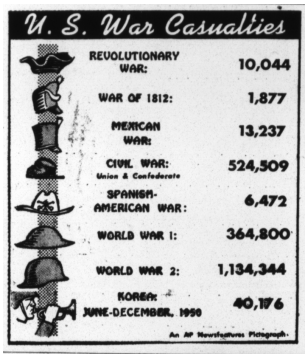


Fig. 4: *An example output from the Newspaper Navigator step.*

2) ***Layer 2: Newspaper Navigator Detection****:* The second layer builds on the cleaned page left from the OpenCV pass and focuses on identifying large non-text components such as photographs, illustrations, and advertisements. This layer uses the pre-trained *Newspaper Navigator* model from the Library of Congress [2], which is based on a fine-tuned Faster R-CNN architecture trained on tens of thousands of annotated newspaper crops. It runs on the remaining portion of the page to detect and segment visual elements that were deliberately left untouched in the first stage.

As shown in Fig. 4, the model effectively localizes complex graphic structures, such as full-width advertisements, comic panels, and other graphic structures, without absorbing neighboring text regions. This stage captures the major visual regions that the OpenCV layer does not identify. The resulting detections mark and crop out the non-text regions, making it easier for later stages to process the remaining content.

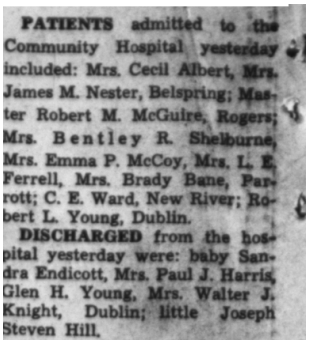
In our observations, a few edge cases include missing faint images or merging several small advertisements into a single bounding box. These issues often appear on lower-quality newspaper scans. Overall, this layer works alongside the OpenCV pass by identifying the page's visual and advertising regions and completing the large-scale layout segmentation. Together with the OpenCV stage, this stage defines the coarse structural map of the newspaper page. The next phase focuses on refining this segmentation through fine-tuned text detection.

3) ***Layer 3: text-only PRIMA Detection Results****:* The fine-tuned *TextOnly-PRIMA* detector showed a clear improvement in both precision and stability compared to the original PubLayNet [19] model. The retrained version produced tighter and more consistent bounding boxes, avoided multi-class overlaps, and preserved column structure even on degraded historical pages. The detector successfully captured dense paragraphs, small captions, and irregular text blocks without excessive merging or fragmentation. Table II presents the quantitative results from evaluation on the `prima_text_dev` split using standard COCO-style [24] metrics.

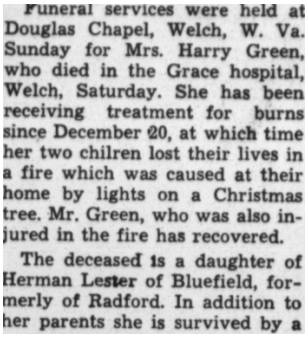
TABLE II: Evaluation of the fine-tuned TextOnly-PRIMA model on the PRImA Layout Analysis dataset (COCO metrics).

Metric	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Score	76.67	95.91	87.48	60.00	81.54	76.10

The results indicate that the model performs well across both loose and strict matching thresholds. A high AP_{50} shows that detected boxes almost always overlap correctly with ground truth regions, while a strong AP_{75} demonstrates precise boundary alignment. The overall mean average precision (mAP) metric reflects balanced performance across varying intersection thresholds. Performance across object sizes (small, medium, and large bounding boxes) remains consistent. Medium and large text blocks are captured reliably, whereas, small regions show good recall despite the noise typical of archival scans.



(a) Detection on a densely printed newspaper page.



(b) Example outputs from the fine-tuned TextOnly-PRIMA detector

Fig. 5: Overall caption for single-column subfigures.

Figures 5a, 5b show the qualitative outputs from the fine-tuned detector. The model captures large paragraphs, multi-line headers, and short captions with minimal overlap or fragmentation, even in degraded or unevenly illuminated pages. It effectively distinguishes text regions from halftone backgrounds and decorative separators, which are common in archival newspapers.

The completeness of text extraction can be seen in Figure 6, which shows the residual image after all detected text regions are masked. The remaining content consists mainly of photographs and illustrations, which were intentionally left untouched in the Newpaper Navigator layer to prevent the loss of nearby text. This outcome confirms that the detector successfully recovers almost all printable text while preserving visual content for later processing. In practice, the TextOnly-PRIMA detector covers roughly 85–90% of the text area, leaving only non-text components to be finalized in the optional reprocessing cycle.

Fig. 6: *Residual image after fine-tuned text detection (TextOnly-PRIMA). Text has been fully captured, leaving only image regions.*

After this stage, and before we move on to visual multi-modal LLM-based (OlmOCR) text extraction, the pipeline re-enters the cycle from the Newspaper Navigator stage to recover any remaining unmapped regions. Running the Navigator and TextOnly-PRIMA layers once more on the residual ensures that small fragments or faint text near image borders are captured before OCR. The second pass finds only a few additional crops, ensuring we don't miss any readable text.

The fine-tuned detector reduces overlapping and ambiguous bounding boxes, which simplifies the information passed to later stages. The manifest files generated at this stage contain a compact list of non-overlapping text regions with precise coordinates and confidence scores. This structured output makes it easier to track provenance and ensures that each OCR crop is well-defined. In practice, this reduces the need for manual review and stabilizes article assembly in later phases. The clean separation of text and non-text achieved here forms the foundation for efficient OCR extraction in Phase 2.

C. *Phase2: Segment-Level Text Extraction.*

The final layer performs OCR on each detected crop using the `allenai/olmOCR-7B-0225-preview` model. Each crop is processed individually with a short verbatim prompt, producing text outputs that are then grouped into articles through column-based adjacency. The prompt used for text extraction is: `Perform text identification of all typed text and punctuation into a single text line verbatim. Provide no explanation and no paragraph numbers.`

```
**RADFORD THEATRE** Dial 2312
**THUR. - FRI.**
THE WHOLE BLAZING STORY OF THE TRISTATE CRIME GANG...and their blood spattered career!
**HIGHWAY 301**
STEVE COCHRAN VIRGINIA GREY GABY ANDRE ANDREW STONE BRYAN FOY
Actual Case From Virginia State Police Files. Filmed Where It Happened.
**VIRGINIAN THEATRE**
**TODAY ONLY**
SCANDAL... ROAMS A RITZY AND FASHIONABLE GIRLS SCHOOL!
**GIRLS' SCHOOL** JOYCE REYNOLDS Ross Ford Laura Elliot Julia Dean Thurston Hall
**FRI. - SAT.**
2 Big Hits & New Super Serial
**"Atom Man vs. Superman"** No. 1
**Whip Wilson** SILVER RAIDERS No. 2
```

Fig. 7: *OCR output for a single advertisement crop, preserving line breaks and emphasis.*

Figure 7 displays a single advertisement crop where the model preserves the original line breaks, spacing, and emphasis marks such as asterisks or all-capital words.

Qualitatively, the transcriptions are clean and readable, with minimal hallucinations or repetition. Multi-column articles are correctly reconstructed, residuals from this phase are minimal, and text coverage typically exceeds 88% of the total text area. The structured `articles.json` and human-readable `articles.txt` outputs provide complete provenance for every detected segment.

VI. Discussion

The experiments confirm that a layered, geometry-aware approach works better for historical newspapers than any single detector running on full pages. Each stage plays a distinct role in improving segmentation quality. The OpenCV layer identifies clean regions without introducing false joins. The Newspaper Navigator stage clears the page of non-text elements such as advertisements and illustrations, reducing confusion for later models. The fine-tuned TextOnly-PRIMA model then recovers remaining text with stable boundaries, even in dense or degraded layouts. Together, these layers isolate readable text with minimal overlap.

This layered design also has practical advantages. Each step can be run independently and tuned or replaced without retraining the entire workflow. The manifest-based structure logs every decision, which allows researchers to reproduce results and trace specific detections back to their source. Compared to the AWS Textract baseline, the open-source ensemble offers higher control, transparency, and low marginal cost at scale, critical for large-scale digitization projects.

Despite these strengths, the system still has limitations. Some decorative borders, thick advertisement frames, and halftone patterns can still trigger false detections even after filtering. The current implementation assumes mostly rectangular layouts and may struggle on pages with curvature, torn edges, or overlapping folds. OCR quality also depends on local print contrast, where faint or broken strokes remain difficult for any vision-language model. However, the modular layout helps isolate such errors, making them easier to diagnose and improve in future iterations.

VII. Conclusion & Future Work

The layered pipeline presented here, OpenCV prefiltering, Newspaper Navigator for non-text structure (with photographs deferred), a TextOnly PRIMA detector, and segment-level OCR converts historical newspaper pages into structured text. The design offers a clearer workflow than a single end-to-end service by separating easy, high-precision steps from harder, model-driven ones, and by recording each decision in manifests that allow stages to be re-run and compared.

We plan to extend the system rather than replace it. We want to add a lightweight classifier that is able to filter out clean pages as the first step. This, in turn, helps us process only clean pages through the pipeline, thereby minimizing errors in detection caused by page quality. This addition will make the workflow more robust across large, mixed collections. We also plan to perform a formal evaluation against benchmark datasets and baseline models to quantify layout accuracy and

OCR consistency across varied sources. Beyond evaluation, we will focus on improving article assembly across columns and pages by linking headlines, paragraphs, and continuations, along with light post-correction for hyphen repair and paragraph merging. We also plan to propagate confidence and uncertainty values from both layout and OCR stages to support quality-aware exports and human review. Finally, we aim to automate the workflow end-to-end, so that new pages can be ingested, processed, and exported with minimal manual intervention. The aforementioned steps will strengthen the pipeline, making it a more reliable and scalable solution to support high volume workflows in digital libraries.

REFERENCES

[1] OpenCV, "OpenCV: Introduction," last accessed Nov 4, 2025. [Online]. Available: https://docs.opencv.org/4.x/d1/dfb/intro.html

[2] LibraryOfCongress, "LibraryOfCongress/newspaper-navigator," May 2020, last accessed Nov 7, 2025. [Online]. Available: https://github.com/LibraryOfCongress/newspaper-navigator

[3] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," 2019, last accessed Nov 6, 2025. [Online]. Available: https://github.com/facebookresearch/detectron2

[4] A. Antonacopoulos, D. Bridson, C. Papadopoulos, and S. Pletschacher, "A realistic dataset for performance evaluation of document layout analysis," in *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR)*, Barcelona, Spain, July 2009, pp. 296–300. [Online]. Available: https://doi.org/10.1109/ICDAR.2009.271

[5] Q. Zhang, B. Wang, V. S.-J. Huang, J. Zhang, Z. Wang, H. Liang, C. He, and W. Zhang, "Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction," 2024. [Online]. Available: https://arxiv.org/abs/2410.21169

[6] S. Najem-Meyer and M. Romanello, "Page layout analysis of text-heavy historical documents: a comparison of textual and visual approaches," 2022. [Online]. Available: https://arxiv.org/abs/2212.13924

[7] W. Zhu, N. Sokhandan, G. Yang, S. Martin, and S. Sathyanarayana, "Docbed: A multi-stage OCR solution for documents with complex layouts," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, 2022, pp. 12 643–12 649. [Online]. Available: https://doi.org/10.1609/aaai.v36i11.21539

[8] D. Fleischhacker, R. Kern, and W. Göderle, "Enhancing OCR in historical documents with complex layouts through machine learning," *International Journal on Digital Libraries*, vol. 26, no. 1, p. 3, Feb. 2025, last accessed Nov 4, 2025. [Online]. Available: https://doi.org/10.1007/s00799-025-00413-z

[9] A. W. Services, "OCR Software, Data Extraction Tool - Amazon Textract - AWS," 2019, last accessed Nov 4, 2025. [Online]. Available: https://aws.amazon.com/textract/

[10] Google, "Tesseract documentation." [Online]. Available: https://tesseract-ocr.github.io/

[11] GoogleAI, "Document AI," last accessed on 6, Nov 2025. [Online]. Available: https://cloud.google.com/document-ai

[12] A. Bawasker, Y. Dedania, V. Karia, N. Bhatt, P. Prajapati, and N. Bhatt, "Enhancing text extraction using OCR with AWS Textract service," in *Advanced Informatics for Computing Research*, ser. CCIS. Springer Nature Switzerland, 2025, pp. 211–220. [Online]. Available: https://doi.org/10.1007/978-3-031-84062-3_17

[13] Nidhi, K. Saluja, A. Mahajan, A. Jadhav, N. Aggarwal, D. Chaurasia, and D. Ghosh, "Table detection and extraction using OpenCV and novel optimization methods," in *2021 International Conference on Computational Performance Evaluation (ComPE)*. IEEE, 2021, pp. 755–760. [Online]. Available: https://doi.org/10.1109/ComPE53109.2021.9752204

[14] T. Grüning, G. Leifert, T. Strauß, J. Michael, and R. Labahn, "A two-stage method for text line detection in historical documents," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 22, no. 3, pp. 285–302, 2019. [Online]. Available: https://doi.org/10.1007/s10032-019-00332-1

[15] S. Ravichandra, S. Siva Sathya, and S. Lourdu Marie Sophie, "Deep learning based document layout analysis on historical documents," in *Advances in Distributed Computing and Machine Learning*. Singapore: Springer Nature Singapore, 2022, pp. 271–281. [Online]. Available: https://doi.org/10.1007/978-981-19-1018-0_23

[16] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, last accessed Nov 4, 2025. [Online]. Available: https://arxiv.org/abs/1804.02767

[17] M. M. B. A. N. Akanda, M. Ahmed, A. S. A. Rabby, and F. Rahman, "Optimum deep learning method for document layout analysis in low resource languages," in *Proceedings of the 2024 ACM Southeast Conference*, ser. ACMSE '24, New York, NY, USA, 2024, p. 199–204. [Online]. Available: https://doi.org/10.1145/3603287.3651184

[18] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023, last accessed Nov 6, 2025. [Online]. Available: https://github.com/ultralytics/ultralytics

[19] Z. Shen, R. Zhang, M. Dell, B. C. G. Lee, J. Carlson, and W. Li, "Layoutparser: A unified toolkit for deep learning based document image analysis," *CoRR*, vol. abs/2103.15348, 2021. [Online]. Available: https://arxiv.org/abs/2103.15348

[20] M. Alberti, M. Bouillon, R. Ingold, and M. Liwicki, "Open evaluation tool for layout analysis of document images," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 43–47. [Online]. Available: https://doi.org/10.1109/ICDAR.2017.311

[21] C. Miller and B. Banerjee, "Optical character recognition for pre-digital historical documents using large language models," in *Proceedings of the 24th IEEE International Conference on Machine Learning and Applications*, Boca Raton, Florida, USA, December 2025, accepted, not yet published. [Online]. Available: https://www.icmla-conference.org/icmla25/ICMLA-Program.pdf

[22] MistralAI, "mistralai/Mistral-Small-3.1-24B-Instruct-2503 · Hugging Face," last accessed on Nov 6, 2025. [Online]. Available: https://huggingface.co/mistralai/Mistral-Small-3.1-24B-Instruct-2503/.

[23] J. Poznanski and et al., "olmOCR: Unlocking Trillions of Tokens in PDFs with Vision Language Models," feb 2025, last accessed on Nov 6, 2025. [Online]. Available: https://huggingface.co/allenai/olmOCR-7B-0225-preview/.

[24] HuggingFace, "Hugging Face Vision: Detection Metrics Demo," last accessed Nov 7, 2025. [Online]. Available: https://huggingface.co/spaces/hf-vision/detection_metrics

[25] VTDLP, "Virginia Tech Digital Libraries | Montgomery Museum," last accessed Nov 4, 2025. [Online]. Available: https://digital.lib.vt.edu/collection/92992h2p

[26] S. Pizer, R. Johnston, J. Ericksen, B. Yankaskas, and K. Muller, "Contrast-limited adaptive histogram equalization: speed and effectiveness," in *[1990] Proceedings of the First Conference on Visualization in Biomedical Computing*, 1990, pp. 337–345. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=109340