

Strategic Research Report: Architectural Evolution and Optimization of LaTeXify for High-Fidelity STEM Document Reconstruction

Executive Summary

The transformation of static Portable Document Format (PDF) files into dynamic, semantically rich LaTeX source code represents one of the most challenging frontiers in document image analysis. This report serves as a comprehensive architectural audit and upgrade roadmap for the LaTeXify repository, designed to elevate it from a basic conversion script to a research-grade system capable of rivaling commercial state-of-the-art (SOTA) solutions like Mathpix. The analysis is predicated on a rigorous review of current literature (2024–2025), specifically targeting the integration of large multimodal models (LMMs) with specialized expert agents.

Our central finding is that a monolithic approach—relying solely on a single vision-language model (VLM) or legacy Optical Character Recognition (OCR) engines—is insufficient for the high-precision demands of Science, Technology, Engineering, and Mathematics (STEM) documents. Complex layouts, nested mathematical notation, and dense tabular data require a **Modular Composite Architecture**. We propose adopting the "Pipeline-of-Experts" design pattern, exemplified by the PDF-Extract-Kit and MinerU frameworks, which orchestrates specialized models for layout detection, formula recognition, and table reconstruction. Furthermore, this report specifically addresses the deployment on next-generation hardware: the NVIDIA RTX 5090 (Blackwell architecture). We identify critical compatibility hurdles with current PyTorch releases and propose a bleeding-edge stack utilizing CUDA 12.8, FP8 quantization via torchao, and vLLM serving to maximize throughput. By implementing the 30 research-backed upgrades and 10 performance optimizations detailed herein, the LaTeXify system will achieve not only semantic accuracy but also visual fidelity verified through novel regression testing methodologies.

Repo Summary: Baseline Assessment and Target State

Current Status: The repository <<https://github.com/vladdiethedecoder/LaTeXify>> serves as the target for this upgrade. While the specific codebase was inaccessible during the reconnaissance phase, we infer a baseline architecture typical of "Generation 1.0" open-source PDF-to-LaTeX tools. These legacy systems predominantly rely on glue-code connecting pytesseract for text extraction and heuristic-based cropping for images, often lacking the semantic understanding required to distinguish between a figure caption and a paragraph, or a matrix and a table.

Inferred Constraints:

- **Dependency on Legacy OCR:** Likely utilization of Tesseract 4/5, which struggles with mixed-language documents and complex mathematical layouts.²
- **Heuristic Layout Analysis:** Reliance on rule-based separators (whitespace detection) rather than learned object detection, leading to failures in multi-column scientific papers.
- **Lack of Hardware Acceleration:** Absence of specialized kernels or mixed-precision support, resulting in high latency on modern GPUs.

Target State (Ambition):

The upgraded LaTeXify must evolve into a Generation 3.0 system. This defines a shift from simple optical character recognition to Document Understanding and Reconstruction.

- **Domain Focus:** Complex STEM literature containing heavy mathematical nomenclature, chemical formulas, and structural data tables.
- **Technology Stack:** A hybrid neuro-symbolic pipeline integrating:
 - **Vision-Language Models (VLMs):** Qwen2.5-VL or InternVL2.5 for holistic understanding and "glue" logic.³
 - **Specialized Expert Models:** UniMERNet for equation syntax⁵ and StructEqTable/TableMaster for grid reconstruction.⁶
 - **Orchestration:** A graph-based execution engine optimizing memory residency on the 32GB RTX 5090.

Key Modules Required:

1. **Layout Detection Agent:** A fine-tuned YOLO-based object detector to segment pages into semantic regions (Header, Body, Formula, Table, Figure).
2. **Formula Recognition Engine:** A specialized encoder-decoder network trained on LaTeX-render pairs to handle complex nested equations.
3. **Table Structure Recognizer:** A model capable of generating HTML or LaTeX tabular code that preserves row/column spans.
4. **Semantic Assembler:** A logic layer that utilizes reading-order algorithms to stitch regions into a coherent .tex document.

Recommended Persona

Primary Persona: The AI Research Systems Architect

This report is authored from the perspective of an AI Research Systems Architect specializing

in Document Intelligence and High-Performance Computing (HPC). This expert bridges the gap between theoretical deep learning research—analyzing arXiv preprints on novel architectures like UniMERNet or FlashAttention-3—and the practical engineering reality of deploying these models on bleeding-edge hardware like the NVIDIA Blackwell architecture. The persona demands rigorous evidence for every architectural decision, favoring solutions that offer verifiable accuracy improvements (BLEU, Edit Distance) and computational efficiency (Tokens/sec, VRAM usage).

Secondary Persona: The MLOps Engineer

Supporting the architect is the MLOps Engineer, focused on the "industrialization" of the research code. This persona ensures that the system is not just a collection of Jupyter notebooks but a robust, reproducible software product. Concerns include containerization (Docker), dependency management (resolving CUDA version conflicts), continuous integration (CI) pipelines for model regression testing, and observability (structured logging/tracing). This dual-perspective approach ensures the advice is both theoretically sound and operationally viable.

Structure & Architecture Review

To achieve the ambitious goal of robust STEM document conversion, the architecture must transition from a linear script to a **Directed Acyclic Graph (DAG)** of specialized processing agents. The industry standard for high-performance document parsing has coalesced around the "Modular Pipeline" approach, as demonstrated by MinerU and PDF-Extract-Kit.⁶

1. Architectural Boundaries: The Pipeline-of-Experts

The system architecture should be delineated into three primary stages: **Perception**, **Extraction**, and **Reconstruction**.

Stage 1: Visual Perception (Layout Analysis)

This is the "eyes" of the system. The input PDF page is converted to a high-resolution image (rasterized) and fed into a Layout Detection Model.

- **Model Recommendation:** **DocLayout-YOLO** or **YOLOv10**.
- **Rationale:** Unlike heavy Transformer-based layout models (e.g., LayoutLMv3), YOLO architectures offer significantly faster inference speeds with comparable accuracy for bounding box detection on standard document elements.⁶
- **Output Classes:** The model must effectively distinguish between:
 - Text (Body paragraphs)
 - Title / Section Header

- Table
- Figure / Chart
- Equation_Inline
- Equation_Display (Block formulas)
- Caption (associated with tables/figures)
- Footer / Header (to be discarded)

Stage 2: Expert Extraction Agents

Once regions are identified, they are routed to specialized models best suited for that data type. This "Mixture of Experts" (MoE) approach prevents the degradation of quality often seen in "Jack-of-all-trades" models.

- **The Math Agent:** Crops identified as Equation are sent to **UniMERNet**.
 - *Why:* UniMERNet consistently outperforms general-purpose VLMs (like Qwen2.5-VL) and older models (Nougat) in complex formula recognition benchmarks (UniMER-Test), specifically handling nested structures and rare symbols with higher fidelity.⁵
- **The Table Agent:** Crops identified as Table are sent to **StructEqTable** or **TableMaster**.
 - *Why:* These models are trained to output structural tokens (HTML or LaTeX delimiters) representing the grid, which is notoriously difficult for standard OCR to reconstruct.⁶
- **The Text/Handwriting Agent:** Text regions are processed by **PaddleOCR** (for speed/standard fonts) or **Qwen2.5-VL** (for handwriting/difficult fonts).
 - *Why:* Qwen2.5-VL has shown near-human performance on handwriting and dense text, making it an ideal fallback for complex scenarios, while PaddleOCR remains the efficiency king for standard printed text.²

Stage 3: Semantic Reconstruction

This stage acts as the "Assembly Line," stitching the disparate outputs back into a cohesive document.

- **Reading Order Logic:** A spatial sorting algorithm (e.g., XY-Cut or a learned topological sort) arranges the text blocks and elements in the correct reading sequence, handling multi-column layouts.⁶
- **Refinement Agent:** A final pass using a Large Language Model (LLM) (e.g., Llama-3 or Qwen2.5-Instruct via vLLM) to repair context-dependent errors (e.g., broken hyphenation across lines, incoherent sentence merges) and ensure LaTeX syntax validity.

2. Orchestration & State Management

Managing the execution of these multiple heavy models on a single RTX 5090 (32GB VRAM) requires sophisticated orchestration.

- **State Object:** Implement a `DocumentState` class. This immutable data structure holds the path to the source PDF, cached raster images, the layout manifest (JSON of bounding boxes), and the intermediate extraction results. This decouples the processing stages, allowing for easier debugging and caching.
- **Resource Management:** A naive script loading YOLO, UniMERNet, StructEqTable, and Qwen-72B simultaneously will trigger an Out-Of-Memory (OOM) error.
 - **Solution:** Adopt a **Client-Server Architecture**. Heavy foundation models (like Qwen2.5-VL) should be served via **vLLM** as a persistent service. The pipeline script acts as a client, making API calls to the local vLLM server for specific tasks, while lighter models (YOLO, UniMERNet) can be loaded/unloaded dynamically or kept in memory if space permits.¹¹

Architecture Diagram (Mermaid)

Code snippet

```
graph TD
    Input --> Preproc
    Preproc --> Layout
    Layout -- "Class: Formula" --> MathAgent
    Layout -- "Class: Table" --> TableAgent
    Layout -- "Class: Text" --> TextAgent
    Layout -- "Class: Figure" --> FigAgent["Figure Captioning Agent (Qwen2.5-VL)"]
    MathAgent --> Assembly
    TableAgent --> Assembly
    TextAgent --> Assembly
    FigAgent --> Assembly
    Assembly --> Order
    Order --> Refine
    Refine --> Output
```

```
subgraph "Orchestration Layer (Ray / Python Async)"
Layout
MathAgent
TableAgent
TextAgent
FigAgent
end

subgraph "Hardware Optimization (RTX 5090)"
vLLM_Server
TextAgent -.-> vLLM_Server
FigAgent -.-> vLLM_Server
Refine -.-> vLLM_Server
end
```

Documentation & DevEx Review

A "research-grade" repository is defined by its reproducibility and ease of use for fellow researchers. The current state likely lacks strict environment definitions, leading to "dependency hell," particularly with the rapid evolution of CUDA and PyTorch versions.

1. Environment & Installation

- **The CUDA 12.8 Requirement:** The NVIDIA RTX 5090 utilizes the Blackwell architecture, which requires CUDA 12.8 and PyTorch 2.7 (or specific nightly builds) for native support.¹³ Standard pip install torch will likely fail or fallback to CPU.
- **Recommendation:** Provide a Dockerfile based on the official NVIDIA container (e.g., `nvcr.io/nvidia/pytorch:25.02-py3` or later).¹⁵ This is the only reliable way to distribute a working environment for such new hardware.
- **Conda Configuration:** Maintain a strict `environment.lock.yml` that pins build versions (e.g., `pytorch-cuda=12.8`).

2. Developer Experience (DevEx)

- **Quickstart:**
 - *User Mode:* pip install latexify-core (providing pre-built wheels for common architectures).
 - *Dev Mode:* make setup (runs Conda env creation, pre-commit installation).

- **Examples:** Include a notebooks/ directory with 01_pipeline_walkthrough.ipynb demonstrating the step-by-step extraction process (visualizing bounding boxes, raw OCR text, and final LaTeX).
 - **Contribution Flow:** Implement pre-commit hooks that run linters (ruff) and static type checkers (mypy). Enforce a "Golden Set" check: no PR can be merged if it degrades performance on a fixed set of 10 test PDFs.
-

Code Quality & Refactoring Suggestions

The transition to a complex pipeline requires rigorous software engineering standards.

1. Module Decoupling

- **Current (Hypothetical):** Monolithic scripts mixing model loading, image processing, and file I/O.
- **Refactor:**
 - src/latexify/models/: Wrappers for neural networks (YOLO, UniMERNet).
 - src/latexify/agents/: Business logic for handling specific content types.
 - src/latexify/pipeline/: Orchestration logic.
 - src/latexify/utils/: Image processing (OpenCV), geometry helpers.

2. Interface Design

- Implement the **Strategy Pattern** for extractors. Define an abstract base class BaseExtractor with a method extract(image: np.ndarray) -> str.
- This allows seamless swapping of backends. For example, a TextExtractor could wrap TesseractExtractor, PaddleOCRExtractor, or QwenVLExtractor. The configuration file determines which is instantiated at runtime.

3. Error Taxonomy

- Move away from generic Python Exceptions. Define a domain-specific error hierarchy:
 - LayoutDetectionError: Model failed to find any content.
 - ModelLoadingError: CUDA OOM or missing weights.
 - CompilationError: Generated LaTeX failed to compile.
 - LowConfidenceError: OCR confidence score below threshold.

4. Secrets & Configuration

- Use hydra for configuration management.⁶ This allows complex, hierarchical configs (e.g., overriding model.math.batch_size from the CLI).
 - Never hardcode paths or API keys. Use .env files loaded via python-dotenv.
-

Output Quality / Runtime Behavior Improvements

The ultimate measure of success is the usability of the generated LaTeX.

1. Auto-Repair & Compilation Loops

- Generated LaTeX often contains syntax errors (unbalanced braces, missing packages).

The system should include a **Compilation Sandbox**.

- **Mechanism:** The system attempts to compile the output using pdflatex or latexmk in a Docker container (for security).
- **Feedback Loop:** If compilation fails, the compiler log and the problematic LaTeX snippet are fed back to the Refinement LLM with a prompt: "Fix the syntax error in this LaTeX code based on this error log."¹⁶

2. Visual Regression Testing

- Textual metrics (Edit Distance, BLEU) do not capture visual layout fidelity.
- **Proposal:** Implement a pipeline using tinyvdiff or custom scripts.¹⁷
 1. Render the generated LaTeX to a PDF.
 2. Convert both the Source PDF and Generated PDF to images.
 3. Compute the **Structural Similarity Index (SSIM)**.
- This metric provides a quantifiable "Visual Fidelity Score" to the user, offering a much stronger guarantee of quality than text matching alone.

3. Observability

- Implement structured logging using structlog. Logs should contain context (e.g., page_num, region_type, model_name, confidence).
- For the RTX 5090, integrate nvmi bindings to log GPU memory usage and temperature during batch processing, helping to diagnose bottlenecks.

Top-30 Research-Backed Upgrades

The following roadmap details 30 distinct, research-backed upgrades divided into five strategic domains. Each recommendation is designed to incrementally move the repository toward the SOTA.

Domain A: Layout & Vision

1. **Upgrade to DocLayout-YOLO:** Replace standard object detectors with **DocLayout-YOLO**.
 - *Why:* Fine-tuned specifically on document datasets (DocStructBench), it offers superior detection of document-specific elements like headers, footers, and side-notes compared to generic YOLO models.⁶
 - *Impact:* High (Fundamental Parsing Accuracy).
2. **Implement XY-Cut / Topological Sorting:** Replace naive top-down sorting.
 - *Why:* Multi-column scientific papers break simple sorting. Recursive XY-Cut algorithms (splitting pages by whitespace channels) reliably reconstruct reading order in complex layouts.⁶
 - *Impact:* High (Readability).
3. **Figure Captioning Agent:** Integrate **Qwen2.5-VL** for figures.

- *Why:* Figures are usually lost in OCR. Qwen2.5-VL can generate semantic descriptions: "A graph showing the relationship between X and Y." This enhances accessibility and searchability.⁴
- *Impact:* Medium (Accessibility).
- 4. **Diagram Vectorization Heuristics:** Attempt to recover vector graphics.
 - *Why:* Rasterizing vector diagrams (PDF drawing commands) loses quality. Detect if a region contains PDF vector paths and attempt to preserve them or convert to SVG/TiKz, though this is high-effort/experimental.
 - *Impact:* Low (Niche visual quality).
- 5. **Smart Cropping with Padding:** Implement context-aware cropping.
 - *Why:* Tight bounding boxes often clip ascenders/descenders of letters or formula indices. Adding dynamic padding (e.g., 5-10% of box height) significantly improves recognition accuracy for downstream agents.
 - *Impact:* Medium (Accuracy).
- 6. **Super-Resolution Pre-processing:** Add an ESRGAN or SwinIR step for low-DPI inputs.
 - *Why:* Scanned PDFs (old papers) suffer from blur. Upscaling before OCR can recover legibility for the layout detector.⁶
 - *Impact:* Medium (Robustness).

Domain B: Mathematical & Scientific Syntax

- 7. **Integrate UniMERNET:** Replace purely generative VLM math extraction with UniMERNET.
 - *Why:* Benchmarks (UniMER-Test) show UniMERNET outperforms larger models like Nougat and Qwen-VL in recognizing complex, nested LaTeX structures.⁵
 - *Impact:* Critical (Math Fidelity).
- 8. **Chemical Equation Detection:** Add a classifier for Chemical formulas.
 - *Why:* Standard math models mangle chemical notation (e.g., H_2O vs $\$H_2O\$$). Detecting these regions allows routing them to a specific prompt ("Convert to `\ce{...}` syntax") or specialized model.⁶
 - *Impact:* Medium (Domain Specificity).
- 9. **Inline Formula Detection via Hybrid OCR:**
 - *Why:* YOLO often misses small inline math ($\$x\$$, $\$\alpha\$$). Use a text recognizer (like Qwen2.5-VL) prompted to be "math-aware," naturally outputting `\(...\)` delimiters within the text stream.³
 - *Impact:* High (Completeness).
- 10. **SymPy Semantic Validation:** Use SymPy to validate formula correctness.
 - *Why:* A generated formula might look correct but be semantically wrong. SymPy's LaTeX parser can check if the generated equation parses into a valid mathematical object, acting as a quality gate.¹⁹
 - *Impact:* Medium (Quality Assurance).

11. Matrix & Array Specialization:

- *Why:* Large matrices are often fragmented. Specialized logic to detect grid-like math structures ensures they are treated as a single `\begin{bmatrix}` block rather than separate lines.
- *Impact:* Medium.

12. Math-Specific Font Recognition:

- *Why:* Distinguishing between \mathcal{L} (Lagrangian) and \mathbb{L} is semantically vital. Training the math agent to respect font styles (mathcal, mathbb) is a key differentiator from basic OCR.
- *Impact:* Low (High-precision Semantic).

Domain C: Tabular & Structured Data

13. Integrate StructEqTable / TableMaster:

- *Why:* These models treat table recognition as a structure prediction task (predicting HTML/LaTeX tags), which handles row/column spans far better than line-detection heuristics.⁶
- *Impact:* High (Table Accuracy).

14. Semantic Table Repair via LLM:

- *Why:* Even structural models make syntax errors. Feeding the raw table content + predicted structure to a small LLM (Llama-3-8B) with the prompt "Fix this LaTeX tabular environment" corrects 90% of syntax errors.²⁰
- *Impact:* Medium (Reliability).

15. Border & Shading Preservation:

- *Why:* Many scientific tables use shading to denote groupings. Ensure the Table Agent supports attributes for `\hline` and `\rowcolor`.
- *Impact:* Low (Visual Fidelity).

16. Table Caption Association:

- *Why:* Layout analysis often separates the table from its caption "Table 1:...". Explicit logic to link these ensures the LaTeX `\caption{}` is correctly placed inside the table environment.
- *Impact:* Medium (Document Structure).

Domain D: Text, Language & Semantics

17. Multilingual OCR with PaddleOCR/Qwen:

- *Why:* Tesseract fails on mixed language docs. PaddleOCR supports 80+ languages robustly. Qwen2.5-VL excels at "in-the-wild" text (handwriting, artistic fonts).²
- *Impact:* High (Versatility).

18. Reference Resolution & Linking:

- *Why:* Convert static `` text into dynamic \cite{ref1} links. Use Regex to extract citation markers and match them against the Bibliography section to generate a .bib file.²¹
- *Impact:* High (Professional Utility).

19. Cross-Page Paragraph Merging:

- *Why:* Pipeline models often treat pages independently, breaking sentences at page boundaries. Logic that checks if a page ends in a non-terminal character and merges it with the next page's start is essential.⁸
- *Impact:* Medium (Flow).

20. Handwriting Mode:

- *Why:* Enabling digitization of lecture notes. A toggle that switches the Text Agent to Qwen2.5-VL (which has superior handwriting recognition) adds significant value for students.²³
- *Impact:* Medium (Feature Set).

21. Auto-Correction of OCR Noise:

- *Why:* OCR often confuses 1 (one) and l (el). A post-processing LLM pass over the text blocks can contextually repair these typos.
- *Impact:* Low (Refinement).

Domain E: Infrastructure & Serving

22. vLLM Integration for Backbone Models:

- *Why:* Running Qwen2.5-VL-72B efficiently requires vLLM's PagedAttention. It supports the RTX 5090's architecture and allows high-throughput serving of the VLM components.¹¹
- *Impact:* Critical (Throughput).

23. FP8 Quantization (Blackwell Specific):

- *Why:* The RTX 5090 supports native FP8 Tensor Cores. Quantizing models to FP8 (via torchao or TransformerEngine) doubles compute throughput and halves memory footprint with negligible accuracy loss.²⁴
- *Impact:* Critical (Performance).

24. Hugging Face Hub Integration:

- *Why:* Seamlessly manage model weights (UniMERNet, Qwen, YOLO). Automatic caching and versioning simplify deployment.²⁷
- *Impact:* Medium (DevEx).

25. NVIDIA Container Toolkit Support:

- *Why:* Essential for reproducing the complex CUDA 12.8 / PyTorch 2.7 environment required for the RTX 5090.¹⁵
- *Impact:* High (Reproducibility).

26. Distributed/Async Orchestration (Ray):

- *Why:* To keep the GPU fed, preprocessing (CPU) and Inference (GPU) must be pipelined. Frameworks like Ray or Python's asyncio can manage this concurrency.
- *Impact:* Medium (Optimization).

27. Structured Logging & Tracing:

- *Why:* Debugging a multi-model pipeline is impossible without traces. Know exactly which agent failed or where latency spiked.
- *Impact:* Medium (Ops).

28. Visual Debugger (Gradio):

- *Why:* A UI that overlays detected bounding boxes on the PDF image allows users (and developers) to instantly diagnose layout detection failures.²⁸
- *Impact:* High (UX/Debug).

29. Hydra Configuration System:

- *Why:* Enables composable, hierarchical configuration, essential for managing the hyperparameters of 4+ different models.⁶
- *Impact:* Medium (DevEx).

30. TinyVDiff Regression Testing:

- *Why:* Automated visual regression testing prevents "silent" failures where text is correct but layout is destroyed.¹⁷
- *Impact:* High (QA).

Top-10 Performance Optimizations for NVIDIA RTX 5090

The NVIDIA RTX 5090 is a "Blackwell" architecture card, representing a significant leap over the previous "Ada Lovelace" generation. It features native FP8 Tensor Cores and improved memory bandwidth. To unlock its potential, one cannot simply run standard PyTorch scripts.

Optimization	Implementation Detail	Expected Benefit	Measurement Strategy
1. FP8 Inference	Use <code>torchao.quantization.quantize_(model, float8_weight_only())</code> . Blackwell supports FP8 (E4M3/E5M2) natively, doubling FLOPS. ²⁶	2x Throughput vs BF16.	Benchmark tokens/sec on Qwen2.5-VL with and without FP8.
2. FlashAttention-3	Install flash-attn (beta) compatible with CUDA 12.8. Configure models to use	1.5-2.0x Speedup in Attention layers.	Profile using nsys (Nsight Systems) to verify flash_fwd_kernel utilization.

	flash_attention_2 (which maps to FA3 on Hopper/Blackwell). ³⁰		
3. vLLM Serving	Deploy the VLM backbone using vLLM with tensor_parallel_size=1 and quantization="fp8". vLLM optimizes memory paging (PagedAttention). ¹¹	3-5x Throughput for generation tasks.	Use vLLM's built-in benchmarking script.
4. Torch Compile	Apply model = torch.compile(model, mode='max-autotune') . This fuses kernels and reduces Python overhead. ³²	20-30% Latency reduction.	Compare end-to-end inference time after warmup iterations.
5. CUDA Graphs	Wrap fixed-size inference (e.g., YOLO, UniMERNet resized crops) in torch.cuda.CUDAGraph (). Eliminates CPU kernel launch latency. ²⁵	Zero CPU Overhead for small batches.	Use torch.autograd.profiler to inspect gaps between kernel executions.
6. Pinned Memory	Ensure DataLoaders use pin_memory=True and num_workers>0. Accelerates data transfer from RAM to VRAM over PCIe.	Faster Host-to-Device copy.	Monitor PCIe bandwidth utilization in nvidia-smi.
7. KV Cache Reuse	Enable enable_prefix_caching=True in vLLM. Efficient for repeated system prompts (e.g., "Convert this image to LaTeX..."). ³³	Reduced Compute for system prompts.	Inspect vLLM logs for cache hit rates.
8. Mixed Precision (AMP)	Use torch.autocast(device_type='cuda', dtype=torch.bfloat16)	Reduced VRAM & Higher Tensor Core usage.	Verify tensor dtypes during the forward pass.

	for layers/models that don't support FP8.		
9. Batching Strategy	Aggregate formula crops from multiple pages into a single batch (e.g., 32 images) before sending to UniMERNet.	Massive Throughput gain vs serial processing.	Monitor GPU Compute utilization (aim for >95%).
10. Aggressive VRAM Mgmt	Implement explicit del model; torch.cuda.empty_cache() hooks or use vLLM's model swapping to manage the 32GB limit effectively.	Prevents OOM crashes.	Track torch.cuda.memory_allocated() logs.

Technical Note on Blackwell Support:

The RTX 5090 (Compute Capability 12.0/sm_120) is bleeding edge. Standard PyTorch distributions (as of early 2025) may not support it fully. You must use PyTorch nightly builds (2.7+) or the NVIDIA NGC containers to ensure the nvcc compiler and CUDA drivers (12.8+) interact correctly with the hardware.¹³ Failing to do so will result in "no kernel image available" errors.

Action Plan & Implementation Roadmap

This roadmap assumes a 7-week sprint to transform the repository.

Phase 1: Foundation & Infrastructure (Weeks 1-2)

- **Owner:** MLOps Engineer.
- **Objective:** Establish a reproducible build environment for the RTX 5090.
- **Key Tasks:**
 - Build the Dockerfile based on nvcr.io/nvidia/pytorch:25.02-py3. Verify CUDA 12.8 operation.
 - Set up the hydra configuration structure.
 - Implement the BaseExtractor interface and refactor existing code into the src/ layout.
- **Milestone:** A "Hello World" pipeline running inside the Docker container on the GPU.

Phase 2: Core Perception & Extraction (Weeks 3-4)

- **Owner:** AI Research Architect.
- **Objective:** Implement the primary "Experts."
- **Key Tasks:**
 - Integrate **DocLayout-YOLO** for page segmentation.

- Integrate **UniMERNet** for formula extraction.
- Integrate **PaddleOCR** for text.
- Develop the **Reading Order** logic (XY-Cut).
- **Milestone:** End-to-end conversion of a text+math PDF with >90% text accuracy.

Phase 3: Advanced Capabilities & Visual Fidelity (Weeks 5-6)

- **Owner:** AI Research Architect.
- **Objective:** Add Table support and Visual Regression testing.
- **Key Tasks:**
 - Integrate **StructEqTable** for table reconstruction.
 - Set up **vLLM** serving for Qwen2.5-VL (Figure captioning/Handwriting).
 - Implement the **Visual Regression Pipeline** (PDF->Image->SSIM).
 - Run benchmarks on the "Golden Set" (OmniDocBench subset).
- **Milestone:** <0.10 Edit Distance on formulas; >0.90 SSIM on rendered pages.

Phase 4: Performance & Release (Week 7)

- **Owner:** Shared.
- **Objective:** Optimization and Documentation.
- **Key Tasks:**
 - Apply **FP8 quantization** and **FlashAttention-3** upgrades.
 - Optimize batch sizes and orchestration logic for the 5090.
 - Finalize documentation (Sphinx) and CI/CD pipelines.
- **Milestone:** Public v3.0 Release.

By adhering to this strategic roadmap, LaTeXify will not only overcome its current limitations but establish itself as a premier, research-grade tool for scientific document processing, fully leveraging the capabilities of next-generation AI hardware.

Works cited

1. Best Open-Source OCR Tools in 2025: A Comparison - Unstract, accessed November 21, 2025, <https://unstract.com/blog/best-opensource-ocr-tools-in-2025/>
2. OCR Hinders RAG: Evaluating the Cascading Impact of OCR on Retrieval-Augmented Generation - arXiv, accessed November 21, 2025, <https://arxiv.org/html/2412.02592v2>
3. Qwen2.5 VL! Qwen2.5 VL! Qwen2.5 VL! | Qwen, accessed November 21, 2025, <https://qwenlm.github.io/blog/qwen2.5-vl/>
4. UniMERNet: A Universal Network for Real-World Mathematical Expression Recognition - GitHub, accessed November 21, 2025, <https://github.com/opendatalab/UniMERNet>
5. opendatalab/PDF-Extract-Kit: A Comprehensive Toolkit for High-Quality PDF Content Extraction - GitHub, accessed November 21, 2025, <https://github.com/opendatalab/PDF-Extract-Kit>
6. A Curated List of Awesome Table Structure Recognition (TSR) Research. - GitHub, accessed November 21, 2025, <https://github.com/MathamPollard/awesome-table-structure-recognition>

7. MinerU2.5: A Decoupled Vision-Language Model for Efficient High-Resolution Document Parsing - arXiv, accessed November 21, 2025, <https://arxiv.org/html/2509.22186v1>
8. Uni-MuMER: Unified Multi-Task Fine-Tuning of Vision-Language Model for Handwritten Mathematical Expression Recognition - arXiv, accessed November 21, 2025, <https://arxiv.org/html/2505.23566v3>
9. 10 Awesome OCR Models for 2025 - KDnuggets, accessed November 21, 2025, <https://www.kdnuggets.com/10-awesome-ocr-models-for-2025>
10. opendatalab/MinerU2.5-2509-1.2B - Hugging Face, accessed November 21, 2025, <https://huggingface.co/opendatalab/MinerU2.5-2509-1.2B>
11. GPT OSS - vLLM Recipes, accessed November 21, 2025, <https://docs.vllm.ai/projects/recipes/en/latest/OpenAI/GPT-OSS.html>
12. Help with RTX 5090 - PyTorch Forums, accessed November 21, 2025, <https://discuss.pytorch.org/t/help-with-rtx-5090/220597>
13. RTX 5090 Training Issues - PyTorch Doesn't Support Blackwell Architecture Yet? - Reddit, accessed November 21, 2025, https://www.reddit.com/r/LocalLLaMA/comments/1law1go/rtx_5090_training_issues_pytorch_doesnt_support/
14. Announcing new VLLM container & 3.5X increase in Gen AI Performance in just 5 weeks of Jetson AGX Thor Launch, accessed November 21, 2025, <https://forums.developer.nvidia.com/t/announcing-new-vllm-container-3-5x-increase-in-gen-ai-performance-in-just-5-weeks-of-jetson-agx-thor-launch/346634>
15. Checking TeX syntax - python - Stack Overflow, accessed November 21, 2025, <https://stackoverflow.com/questions/44442265/checking-tex-syntax>
16. tinyvdiff: Minimalist visual regression testing plugin for pytest - Nan Xiao, accessed November 21, 2025, <https://nanx.me/blog/post/tinyvdiff/>
17. Measuring similarity in two images using Python | by Param Raval | TDS Archive - Medium, accessed November 21, 2025, <https://medium.com/data-science/measuring-similarity-in-two-images-using-python-b72233eb53c6>
18. How to compare two LaTeX math strings for equality - TeX, accessed November 21, 2025, <https://tex.stackexchange.com/questions/425001/how-to-compare-two-latex-math-strings-for-equality>
19. Beyond Isolated Dots: Benchmarking Structured Table Construction as Deep Knowledge Extraction - arXiv, accessed November 21, 2025, <https://arxiv.org/html/2507.16271v1>
20. PDF to LaTeX Converter: Best Tools and Techniques - Underleaf, accessed November 21, 2025, <https://www.underleaf.ai/blog/pdf-to-latex-converter-guide>
21. (PDF) unarXive: a large scholarly data set with publications' full-text, annotated in-text citations, and links to metadata - ResearchGate, accessed November 21, 2025, https://www.researchgate.net/publication/339628509_unarXive_a_large_scholarly_data_set_with_publications_full-text_annotated_in-text_citations_and_links_to_metadata

22. Benchmarking vision-language models on OCR in dynamic video environments | Hacker News, accessed November 21, 2025,
<https://news.ycombinator.com/item?id=43045801>
23. Using FP8 and FP4 with Transformer Engine - NVIDIA Docs Hub, accessed November 21, 2025,
https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/fp8_primer.html
24. Accelerating Llama3 FP8 Inference with Triton Kernels - PyTorch, accessed November 21, 2025, <https://pytorch.org/blog/accelerating-llama3/>
25. Releases · pytorch/ao - GitHub, accessed November 21, 2025,
<https://github.com/pytorch/ao/releases>
26. GOT-OCR2 - Hugging Face, accessed November 21, 2025,
https://huggingface.co/docs/transformers/v4.49.0/en/model_doc/got_ocr2
27. Quick Usage - MinerU, accessed November 21, 2025,
https://opendatalab.github.io/MinerU/usage/quick_usage/
28. PyTorch now offers native quantized variants of popular models! : r/LocalLLaMA - Reddit, accessed November 21, 2025,
https://www.reddit.com/r/LocalLLaMA/comments/1nlguk9/pytorch_now_offers_native_quantized_variants_of/
29. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision - arXiv, accessed November 21, 2025,
<https://arxiv.org/html/2407.08608v2>
30. SageAttention3: Microscaling FP4 Attention for Inference and An Exploration of 8-bit Training - arXiv, accessed November 21, 2025,
<https://arxiv.org/html/2505.11594v2>
31. PyTorch 2.7 Release, accessed November 21, 2025,
<https://pytorch.org/blog/pytorch-2-7/>
32. Blackwell FP8 W8A8 NVFP4 support discussion : r/LocalLLaMA - Reddit, accessed November 21, 2025,
https://www.reddit.com/r/LocalLLaMA/comments/1lx4zpr/blackwell_fp8_w8a8_nvfp4_support_discussion/