## Document Log: Approaches tried for Advanced AI OCR System Development

**Objective**

The goal is to build an advanced AI-based OCR system that can accurately extract heterogeneous data (text, tables, images, charts, etc.) from PDFs.

**Approach 1: OCR-Based Extraction (Initial Attempt)**

**Tools Used:**

* Tesseract OCR
* OpenCV for preprocessing
* PDFplumber for text extraction
* PyMuPDF for PDF processing

**Steps:**

1. **Preprocessing:** Used OpenCV to enhance image quality by applying thresholding, denoising, and contrast adjustments.
2. **Text Extraction:** Used Tesseract OCR with different configurations (--oem 3 --psm 6).
3. **Table Extraction:** Used PDFplumber and PyMuPDF to detect and extract tables.
4. **Image Extraction:** Used PyMuPDF to extract images and store them separately.

**Challenges & Failures:**

* Inconsistent table structure extraction.
* Heavy dependency on image quality; noisy PDFs yielded poor results.

**Approach 2: LLaMA 3.2 Vision API for Document Chunking & Extraction**

**Tools Used:**

* LLaMA 3.2 Vision API
* LangChain for document processing
* Semantic chunking for better extraction

**Steps:**

1. **Chunking Approach:** Instead of extracting page-by-page, applied semantic chunking.
2. **Vision API Processing:** Sent chunks to the LLaMA 3.2 Vision API for structured extraction.

**Challenges & Failures:**

* **API cost and latency:** API calls were slow for large PDFs and ended up in rate limit errors even after chunking.

**Approach 3: Exploring Landing AI's Agentic Document Extraction**

**Tools Used:**

* Landing AI’s Agentic Document Extraction
* NLP-based structuring techniques

**Steps:**

1. **Tried automated extraction using Landing AI’s approach.**
2. **Compared with OCR-based and Vision API-based methods.**

**Challenges & Failures:**

* **Limited documentation and customization:** Difficult to fine-tune for specific document structures.
* **Integration issues with local processing:** Required more cloud dependency than desired.

**Approach 4: Fully Local AI OCR with Ollama & Table Transformer**

**Tools Used:**

* Ollama for local LLM inference
* Table Transformer for table extraction
* Image segmentation for structured data extraction
* LangChain for text preprocessing

**Steps:**

1. **Set up Ollama for local LLM inference.**
2. **Used Table Transformer for improved table recognition.**
3. **Integrated text, table, and image extraction into a structured format.**

**Challenges & Failures:**

* **Ollama performance tuning required:** The inference was slow initially.
* **Table extraction inconsistencies:** Some tables were detected incorrectly.
* **JSON serialization errors:** Encountered serialization issues while structuring the extracted data.

**Next Steps & Planned Improvements**

1. **Optimize Ollama’s inference speed for local processing.**
2. **Refine preprocessing techniques to enhance OCR accuracy.**
3. **Fix JSON serialization issues to maintain structured output.**
4. **Improve semantic chunking before sending data to LLMs for better accuracy.**

This document summarizes all attempted methodologies, highlighting successes and failures. Future iterations will focus on refining preprocessing, local LLM inference, and structured output formatting.