Table 1: Versions and configuration options considered for each tested asset. * denotes the default setting.

Asset	Version	OCR	Layout	Tables
Docling	2.5.2	EasyOCR*	default	TableFormer (fast)*
Marker	0.3.10	Surya*	default	default
MinerU	0.9.3	auto*	doclayout_yolo	rapid_table*
Unstructured	0.16.5	`hi_res` with table structure		

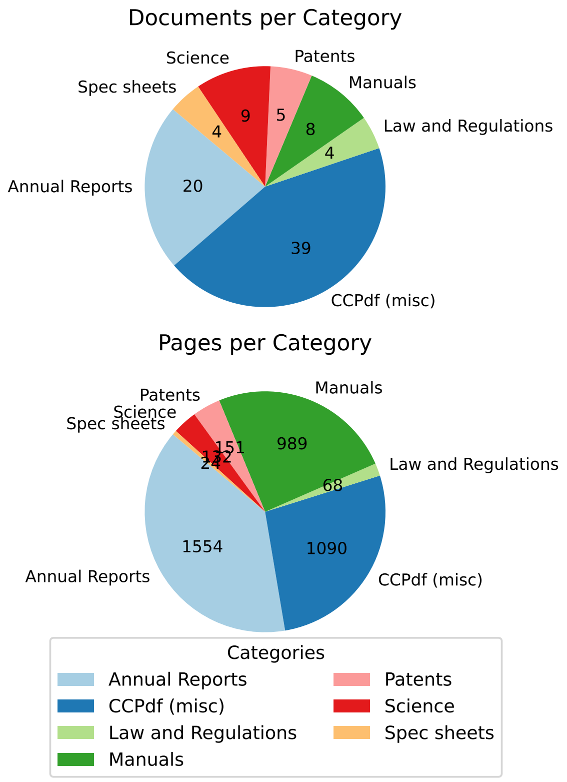

Figure 2: Dataset categories and sample counts for documents and pages.

5.3 Benchmarking Methodology

We implemented several measures to enable a fair and reproducible benchmark across all tested assets. Specifically, the experimental setup accounts for the following factors:

- All assets are installed in the latest available versions, in a clean Python environment, and configured to use the state-of-the-art processing options and models, where applicable. We selectively disabled non-essential functionalities to achieve a compatible feature-set across all compared libraries.
- When running experiments on CPU, we inform all assets of the desired CPU thread budget of 8 threads, via the `OMP_NUM_THREADS` environment variable and any accepted configuration options. The L4 GPU on our AWS EC2 VM is hidden.
- When running experiments on the L4 GPU, we enable CUDA acceleration in all accepted configuration options, ensure the GPU is visible and all required runtimes for AI inference are installed with CUDA support.

Table 1 provides an overview of the versions and configuration options we considered for each asset.

5.4 Results

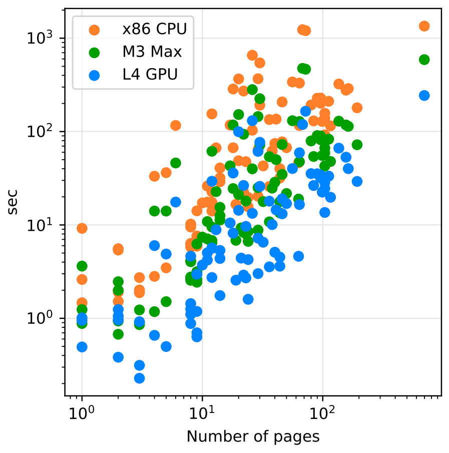

Figure 3: Distribution of conversion times for all documents, ordered by number of pages in a document, on all system configurations. Every dot represents one document. Log/log scale is used to even the spacing, since both number of pages and conversion times have long-tail distributions.

Runtime Characteristics To analyze Docling's runtime characteristics, we begin by exploring the relationship between document length (in pages) and conversion time. As shown in Figure 3, this relationship is not strictly linear, as documents differ in their frequency of tables and bitmap elements (i.e., scanned content). This requires OCR or table structure recognition models to engage dynamically when layout analysis has detected such elements.

By breaking down the runtimes to a page level, we receive a more intuitive measure for the conversion speed (see also Figure 4). Processing a page in our benchmark dataset requires between 0.6 sec (5th percentile) and 16.3 sec (95th percentile), with a median of 0.79 sec on the x86 CPU. On the M3 Max SoC, it achieves 0.26/0.32/6.48 seconds per page (.05/median/.95), and on the Nvidia L4 GPU it achieves 57/114/2081 milliseconds per page (.05/median/.95). The