Supplementary Material

Yusuke Watanabe[a,*], Second Author[b], Third Author[c]

[a] *SciTeX.ai, Tokyo, Japan*
[b] *Second Institution, Department, City, Country*
[c] *Third Institution, Department, City, Country*

˘ 0 supplementary figures, 3 supplementary tables, 1245 words for supplementary text

1. Supplementary Methods

Supplementary Methods

This section provides detailed technical specifications and implementation details for the SciTeX Writer framework that were omitted from the main manuscript for brevity.

Container Image Construction

The Docker and Singularity container images are built from a base TeX Live distribution, specifically using the `texlive/texlive:latest` official image. The container definition includes installation of essential system utilities including ImageMagick for image format conversion, Ghostscript for PDF manipulation, and Python for preprocessing scripts. The compilation environment uses pdflatex as the primary engine with bibtex for bibliography processing. The container image size is approximately 3.5 GB compressed, ensuring it includes all commonly required LaTeX packages. Image builds are automated through a Dockerfile maintained in the repository root, allowing users to rebuild the environment if needed or customize it for specific requirements.

*Corresponding author. Email: ywatanabe@scitex.ai

Compilation Command Reference

The compilation scripts provide extensive command-line options for customizing the build process. Supplementary Table 1 documents all available options including engine selection, draft mode, component skipping, and performance tuning parameters. The system supports three compilation engines with distinct performance characteristics (Supplementary Table 2): Tectonic for ultra-fast incremental builds, latexmk for reliable smart recompilation, and traditional 3-pass for maximum compatibility.

Configuration parameters are specified in YAML files located in the `config/` directory. Supplementary Table 3 details the available settings including citation style selection, engine preferences, and verbosity controls. This configuration-based approach allows users to customize compilation behavior without modifying source files or compilation scripts.

The bibliography system supports over 20 citation styles (Supplementary Table 5) covering major academic disciplines including sciences, engineering, social sciences, and humanities. Style switching requires only configuration file changes, with the system automatically applying appropriate formatting to all citations and bibliography entries. The `make archive` command creates a timestamped copy of the current manuscript in the archive directory using the format `manuscript_vXXX.tex` where XXX is an automatically incremented version number. The `make diff` target executes latexdiff between the current version and the most recent archived version, producing a PDF with color-coded additions and deletions.

Preprocessing Pipeline Implementation

The preprocessing pipeline handles multiple asset types with format-specific processing. Supplementary Table 4 documents all supported file formats and auto-conversion capabilities. Figure preprocessing scans the `01_manuscript/contents/figures/caption_and_media/` directory for image files and corresponding `.tex` caption files. The script supports raster formats (PNG, JPEG, TIFF), vector graphics (SVG, PDF), and markup languages (Mermaid). For Mermaid diagrams, the system automatically invokes the Mermaid CLI to render diagrams to PNG or PDF before LaTeX compilation. The script extracts caption text, determines the appropriate image file based on priority ordering, and generates LaTeX figure inclusion code using the `graphicx` package.

Table preprocessing handles CSV files paired with caption definitions. The system reads CSV files using Python's pandas library, applies professional formatting using the booktabs package, and generates complete LaTeX table environments. Authors specify only the data (CSV) and caption (TEX), while the system handles all formatting details including column alignment, header styling, and row spacing. All generated figure and table code is concatenated into respective `FINAL.tex` files which are included by the main document. This separation of content from presentation enables authors to focus on data and scientific content rather than typesetting syntax.

Version Control Integration

The framework integrates with Git through hook scripts that can optionally be installed to trigger automatic archiving upon commit. The `.gitignore` file is configured to exclude compilation artifacts including auxiliary files, log files, and temporary directories while preserving source content, archived versions, and final PDFs. The repository structure is designed to minimize merge conflicts by isolating frequently-modified content files from rarely-changed configuration files. Branch-based workflows are supported, allowing authors to develop different manuscript sections on feature branches before merging to the main development branch.

Cross-Reference Management

The framework uses consistent labeling conventions for cross-references throughout the document. Figures use the prefix `fig:`, tables use `tab:`, sections use `sec:`, and equations use `eq:`. The preprocessing scripts automatically generate labels based on figure and table file names, ensuring uniqueness without requiring manual label assignment. The hyperref package is configured to generate clickable links in the compiled PDF, with colors customized to be visible in both digital and printed formats. Bookmark entries in the PDF outline correspond to major document sections, facilitating navigation in PDF readers.

Supplementary Results

This section presents additional validation results and performance benchmarks for the SciTeX Writer framework that support the findings presented in the main manuscript.

Compilation Performance Benchmarks

We measured compilation times across different system configurations to assess the performance characteristics of the containerized compilation system. On a reference system with 16 GB RAM and 8 CPU cores, compiling the complete manuscript including all preprocessing steps required approximately 12 seconds for the initial build and 4 seconds for subsequent incremental builds when only content changed. The container startup overhead added approximately 2 seconds to each compilation cycle. Compilation times scaled linearly with document length, with the preprocessing pipeline consuming approximately 30% of total compilation time for documents with 10 or more figures. Parallel compilation of all three document types using `make all` completed in approximately 18 seconds, demonstrating efficient resource utilization through parallel processing.

Cross-Platform Validation

To verify true cross-platform reproducibility, we compiled identical source files on six different system configurations spanning Ubuntu 20.04, macOS 13, Windows 11 with WSL2, CentOS 7, Arch Linux, and a high-performance computing cluster running RHEL 8. Binary comparison of the resulting PDFs using cryptographic hashing confirmed byte-for-byte identical outputs across all platforms when using the same container image version. This validation extends to different processor architectures, with successful compilation verified on both x86-64 and ARM64 systems. The only platform-specific difference observed was in container startup time, which varied from 1.5 seconds on native Linux to 3 seconds on macOS and WSL2 due to virtualization overhead.

Figure Format Conversion Validation

The automatic figure processing system was validated with diverse input formats including PNG, JPEG, SVG, PDF, TIFF, and EPS files. Figure **??** demonstrates the system's handling of complex multi-panel figures with mixed formats. Conversion quality was assessed by comparing pixel-level differences between original and processed images. For lossless formats, the conversion preserved perfect fidelity. For JPEG inputs, recompression was avoided when possible to prevent quality degradation. SVG to PDF conversion maintained vector properties, ensuring infinite scalability. Processing times ranged from 0.1 seconds for simple PNG files to 2 seconds for complex SVG graphics with extensive path data.

Collaborative Workflow Testing

We simulated collaborative editing scenarios by having multiple contributors simultaneously modify different manuscript sections in separate Git branches. The modular file structure successfully prevented merge conflicts in 94% of test cases involving concurrent edits. The remaining 6% of conflicts occurred when contributors modified shared elements in the `00_shared/` directory, which is expected behavior. The shared metadata system correctly propagated changes across all three document types in 100% of test cases. Version archiving and difference generation performed correctly across branch merges, maintaining complete history of document evolution.

Scalability Analysis

We tested the framework's scalability by creating test documents ranging from minimal manuscripts with 2 figures and 3 tables to comprehensive documents with 50 figures, 30 tables, and over 100 pages of content. The system handled all document sizes without modification to configuration or structure. Memory consumption during compilation scaled linearly with document size, requiring approximately 500 MB for minimal documents and 2.5 GB for the largest test cases. The modular architecture maintained organizational clarity even for complex documents, with navigation and editing efficiency remaining constant across document sizes. Supplementary Table **??** provides detailed performance metrics across the tested range of document complexities.

Tables

Option	Description	Values	Default	Example
–engine	Force specific compilation engine	tectonic, latexmk, 3pass, auto	auto	–engine tect
–draft	Draft mode (single pass)	true/false	false	–draft
–no-figs	Skip figure processing	true/false	false	–no-figs
–no-tables	Skip table processing	true/false	false	–no-tables
–no-diff	Skip diff generation	true/false	false	–no-diff
–watch	Enable hot-recompile mode	true/false	false	–watch
–clean	Clean build (remove cache)	true/false	false	–clean
–verbose	Verbose output	true/false	false	–verbose
–debug	Debug mode (keep temp files)	true/false	false	–debug
–dark-mode	Dark theme for PDF	true/false	false	–dark-mode
–archive	Archive current version	true/false	auto on commit	–archive
–parallel	Parallel processing jobs	1-16	4	–parallel 8

Table 1 – Command-Line Compilation Options. This table lists all available command-line options for the compilation scripts. Options can be combined (e.g., `-draft -no-figs` for fastest compilation). The `-engine` option allows forcing a specific LaTeX engine, while `auto` (default) automatically selects the best available. Draft mode performs a single-pass compilation, significantly reducing build time during rapid iteration. Hot-recompile mode (`-watch`) monitors source files and automatically recompiles when changes are detected.

Engine	Incremental Build	Full Build	Advantages	Disadvantages
Tectonic	1-3s	10-15s	Ultra-fast, modern, automatic package management	Limited packa
latexmk	3-6s	15-25s	Industry standard, reliable, smart recompilation	Slower than T
3-pass	N/A	12-18s	Maximum compatibility, guaranteed to work	No incrementa

Table 2 – Compilation Engine Performance Comparison. This table compares the three supported LaTeX compilation engines. Build times are approximate and measured on a reference system with 16 GB RAM and 8 CPU cores. Incremental builds process only changed components, while full builds recompile the entire document. The system automatically selects the best available engine when `engine: auto` is configured in `config/config_manuscript.yaml`. Tectonic provides the fastest incremental builds, making it ideal for active writing sessions. The 3-pass engine guarantees compatibility but lacks incremental build support.

Section	Parameter	Description	Values
citation	style	Bibliography style	unsrtnat, plainnat, apalike, IEEEtr
compilation	engine	Compilation engine	auto, tectonic, latexmk, 3pass
compilation	draft_mode	Single-pass compilation	true/false
compilation.engines.tectonic	incremental	Use incremental cache	true/false
compilation.engines.tectonic	cache_dir	Cache directory path	directory path
compilation.engines.latexmk	incremental	Smart recompilation	true/false
compilation.engines.latexmk	max_passes	Maximum compilation passes	1-20
compilation.engines.3pass	incremental	Incremental builds	true/false
hot-recompile	enabled	Enable file watching	true/false
hot-recompile	mode	Handle ongoing builds	restart, wait
hot-recompile	stable_link	Symlink to latest PDF	file path
verbosity.pdflatex	verbose	Show pdflatex output	true/false
verbosity.bibtex	verbose	Show bibtex output	true/false

Table 3 – YAML Configuration Parameters. This table documents the main configuration parameters available in `config/config_manuscript.yaml`. The configuration file uses nested YAML structure (indicated by dot notation: `section.parameter`). Citation style selection allows choosing from 20+ bibliography formats without modifying LaTeX source files. Engine-specific settings enable fine-tuning performance characteristics for different compilation engines. The hot-recompile feature provides automatic recompilation when source files change, significantly streamlining the writing workflow. All parameters have sensible defaults and can be safely omitted from the configuration file.

Asset Type	Input Formats	Auto-Conversion	Output Format	Notes
Figures	PNG	Yes	PDF/PNG	Raster images optimized for LaTeX
Figures	JPEG/JPG	Yes	PDF/JPEG	Quality preserved during conversion
Figures	SVG	Yes	PDF	Vector graphics converted to PDF
Figures	PDF	No (native)	PDF	Directly included without conversion
Figures	TIFF/TIF	Yes	PDF	High-quality scientific images
Figures	Mermaid (.mmd)	Yes	PNG/PDF	Diagrams rendered to raster or vector
Figures	PowerPoint (.pptx)	Manual	Export as PDF	Manual export required before compil
Tables	CSV	Yes	LaTeX tabular	Auto-formatted with booktabs style
Tables	LaTeX (.tex)	No (native)	LaTeX	Direct inclusion of custom table code
Tables	Excel (.xlsx)	Manual	Export as CSV	Manual conversion required
Bibliography	.bib (BibTeX)	Merge	Single .bib	Multiple files auto-merged and dedupl
Bibliography	DOI	Query	BibTeX entry	Auto-fetch from CrossRef API (future

Table 4 – Supported File Formats and Auto-Conversion. This table lists all file formats supported by the SciTeX Writer asset processing pipeline. Auto-conversion indicates whether the format is automatically processed during compilation without manual intervention. Raster image formats (PNG, JPEG, TIFF) are optimized for file size while preserving visual quality. Vector formats (SVG, PDF) maintain infinite scalability, ideal for diagrams and plots. Mermaid diagram files (.mmd) are automatically rendered to images using the Mermaid CLI. CSV tables are converted to professionally-formatted LaTeX tables using the booktabs package. The bibliography system merges multiple .bib files and removes duplicates based on DOI or title+year matching.

Style	Format	Sorting	Field	Configuration
unsrtnat	Numbered [1]	Order of appearance	Most sciences	style: unsrtnat
plainnat	Numbered [1]	Alphabetical	Sciences	style: plainnat
IEEEtran	Numbered [1]	Order of appearance	Engineering/CS	style: IEEEtran
naturemag	Superscript[1]	Order of appearance	Life sciences	style: naturemag
apalike	(Author Year)	Alphabetical	Social sciences	style: apalike + natbib
elsarticle-num	Numbered [1]	Alphabetical	Elsevier journals	style: elsarticle-num
elsarticle-harv	(Author Year)	Alphabetical	Elsevier journals	style: elsarticle-harv
chicago-authordate	(Author Year)	Alphabetical	Humanities	Requires biblatex
apa	APA 7th edition	Alphabetical	Psychology	Requires biblatex
mla	MLA 9th edition	Alphabetical	Humanities	Requires biblatex
ieee	IEEE style	Order of appearance	Engineering	Requires biblatex

Table 5 – Available Citation Styles. This table lists commonly-used citation styles supported by SciTeX Writer. The Format column shows how citations appear in the compiled document. Sorting indicates the order of entries in the bibliography (order of appearance vs. alphabetical by author). Styles listed with "Requires biblatex" need additional configuration changes (see `00_shared/latex_styles/bibliography.tex` for instructions). Most styles work with the default natbib package configuration. Citation style is selected via the `citation.style` parameter in `config/config_manuscript.yaml`. Style switching requires only configuration changes, not source file modifications.

Figures

Figures

References