# Inconsistencies in TEX-produced PDF Documents

Anonymous Author(s)

## ABSTRACT

TEX is a widely used typesetting system adopted by most publishers and professional societies. While TEX is responsible for generating a significant number of documents, irregularities in the TEX ecosystem may produce inconsistent documents. These inconsistencies may occur across different TEX engines or different versions of TEX distributions, resulting in failures to adhere to formatting specifications, or the same document rendering differently for different authors. In this work, we investigate and quantify the robustness of the TEX ecosystem through a large-scale study of 432 documents. We developed an automated pipeline to evaluate the cross-engine and cross-version compatibility of the TEX ecosystem. We found significant inconsistencies in the outputs of different TEX engines: only 0.2% of documents compiled to identical output with XƎTEX and pdfTEX due to a lack of cross-engine support in popular LATEX packages and classes used in academic conferences. A smaller—yet significant—extent of inconsistencies was found across different TEX Live distributions, with only 42.1% of documents producing the same output from 2020 to 2023. From a sample of 10 unique root causes of inconsistencies, we identified two new bugs in LATEX packages and five existing bugs that were fixed independently of this study. We also observed potentially unintended inconsistencies across different TEX Live distributions beyond the updates listed in changelogs. We expect that this study will help authors of TEX documents to avoid unexpected outcomes by understanding how they may be affected by the often undocumented subtleties of the TEX ecosystem, while benefiting developers by demonstrating how different implementations result in unintended inconsistencies.

## KEYWORDS

TEX, LATEX, typesetting, PDF documents

**ACM Reference Format:**
Anonymous Author(s). 2024. Inconsistencies in TEX-produced PDF Documents. In *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2024)*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

TEX is a typesetting system widely used to produce output of the highest typographic quality. LATEX is a set of macros built on top of TEX; LATEX systems are currently the dominant typesetting environment used in scientific publishing [37], and have been adopted by most publishers and professional societies as their preferred format [10, 24]. Overleaf, an online LATEX editor, amassed 11 million users in 2022 [2]. Evidently, the TEX ecosystem is responsible for generating a large amount of documents.

Bugs and inconsistencies in the TEX ecosystem may produce documents that are incorrect or inconsistent, thus affecting the communication of information. Given a source TEX file, the output of each TEX engine is usually expected to be visually identical. Unexpected differences in output between different engines compromises the quality of typeset documents. Examples of such differences include information loss or incorrect formatting. Differences in output are highly undesirable when preparing documents that must adhere to a strict specification, such as in academic conferences or journals where violating formatting requirements results in desk rejection.

While authors may expect the compiled output of their TEX files to remain consistent across different TEX engines and different versions of TEX Live, this hypothesis has not been systematically studied. To our knowledge, there is currently no existing work on a systematic comparison of the output produced by different TEX engines and TEX Live distributions. While release notes and documentation are helpful, dependencies on various LATEX macros and simultaneous updates of different packages make it difficult for authors to determine how they might be affected by different engines or TEX Live versions respectively.

Motivated by the limited information available for authors to navigate the complex TEX ecosystem, we pose and answer the following research questions (RQs):

**RQ1** *How interchangeable are different TEX engines?*
Knowing this helps authors make informed choices between TEX engines; understand how their choice of engine may affect their writing experience and output; and avoid unexpected outcomes when their documents are compiled with different engines. While each TEX engine may list its unique features, implementation differences might introduce unintended inconsistencies that are of interest to authors. Since the PDFs produced by different engines may inevitably contain minor differences that are transparent to end users (such as in differences in metadata), we consider outputs *consistent* if they are visually identical. While different TEX engines are expected to produce consistent output when given the same input file, we hypothesise that this may not always be the case due to differences in their implementations.

**RQ2** *How interchangeable are different versions of TEX Live?*
Knowing this helps authors of TEX files to understand the potential consequences of upgrading—or failing to upgrade—their TEX distributions. Characterising the differences observed in this study concretises the inconsistencies that are of interest to authors. Authors may thus make an informed decision on when to upgrade their TEX-related software or which versions to use to ensure that their documents meet given specifications.

**RQ3** *Are inconsistencies across TEX Live versions considered bugs?*
Knowing this helps developers of TEX-related software (like LATEX packages) to understand how changes in behaviour are triaged

*ISSTA 2024, 16-20 September, 2024, Vienna, Austria*
© 2024 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

and what extent of inconsistency is considered reasonable between versions. Authors may also benefit by knowing what to expect from, and how they may be affected by, updating their TeX Live distribution. We chose to focus on TeX Live versions over TeX engines as the subtle differences between engines and the lack of a formal specification make it difficult to define expected behaviour in each engine.

To answer these questions, we conducted a large-scale empirical study on the inconsistencies in the TeX ecosystem. We studied the differences between the outputs of three different TeX engines as well as four different versions of TeX Live distributions when compiling the same source files from 432 different documents, automating the comparison to identify common differences between engines. We additionally used various metrics such as pixel-wise differences and textual differences to measure "interchangeability" by characterising these differences in a manner accessible for authors and users of the TeX ecosystem. In summary, we found:

- Several incompatibilities of LaTeX classes with different TeX engines. These incompatibilities range from compilation errors to significant differences in the output PDF. Several popular LaTeX classes, such as document classes for conference submissions, were compatible only with pdfTeX and incompatible with other engines.
- A large extent of differences between TeX engines: only 0.2% of documents had identical (pixel-wise) output with pdfTeX and XeTeX, and only 1.4% had identical output with LuaTeX and XeTeX.
- A smaller, but still significant, extent of differences between different TeX Live distributions: only 42.1% of documents compiled to the same output across all four releases. The change from TeX Live 2020 to 2021 was the most significant, with 27.1% of documents producing different output across these releases but consistent output in subsequent releases.
- Two new bugs, and five existing bugs, in LaTeX packages and classes. Other inconsistencies found were considered expected differences, with varying levels of documentation.
- Undocumented changes in the output of popular document classes over different versions of TeX Live, highlighting the importance of verifying the TeX Live version used in compiling documents that must adhere to strict specifications.
- Minor differences (such as the spacing between letters differing by a few pixels) were common. Although these inconsistencies might not affect the information conveyed in the document, multiple inconsistencies may culminate in larger and significant differences (such as a different number of pages in the output document).

All data, scripts, and results are available at https://zenodo.org/records/10778054. To our knowledge, this large-scale study is the first such study on the inconsistencies of the TeX ecosystem. The inconsistencies we have uncovered provide a strong starting point for developing test cases under more controlled environments. Developers within the TeX ecosystem (such as developers of TeX packages) may also gain a better awareness of how to develop packages compatible with all engines to increase their user base. Additionally, users of the TeX ecosystem may gain a better understanding of which TeX engine will best suit their needs, and how the TeX engine they choose to use may affect their output document. Finally, by understanding how the different implementations of TeX engines result in unintended—and possibly significant—side effects, both users and developers will benefit from an enhanced understanding of the ecosystem as a whole and how it affects the processes and results of typesetting.

## 2 BACKGROUND

*TeX Live.* TeX Live is a software distribution that provides a comprehensive TeX system including various packages, macros, and engines [17]. TeX Live is released yearly, and while it is possible to update a TeX distribution between official releases, it is *"not necessary or particularly recommended"* especially since the set of packages and programs in the official release is *"the only one which gets tested as a coherent release"* [15].

*TeX engines.* Since TeX's release in 1978, other engines have been developed to extend the functionality of the original TeX program. This paper discusses three of these engines, all derived from TeX:

(1) pdfTeX, and its LaTeX variant pdfLaTeX, is a backward-compatible extension of TeX that can directly output PDF ("Portable Document Format") files from TeX files [33].
(2) XeTeX is a Unicode TeX engine that enables convenient use of OpenType fonts and multilingual typesetting. It is not backward-compatible with TeX [42].
(3) LuaTeX introduces a Lua interpreter, allowing authors to integrate Lua code into their macros. It is not backward-compatible with TeX [39].

*PDF outputs.* Given an input file, TeX aims to be system-agnostic in generating output: input files should *"generate the same output on any system on which they were processed—same hyphenation, same line breaks, same page breaks, same everything"* [4]. While different TeX engines are not required to produce the same output given the same input file, these three engines are compatible with the LaTeX format, and thus should produce relatively similar output according to this format. Engines like LuaTeX *"may be used as a drop-in replacement"* most of the time; for both LuaTeX and XeTeX, due to the interfaces provided by the LaTeX format, *"for most LaTeX end users, these subtleties are transparent"* [1].

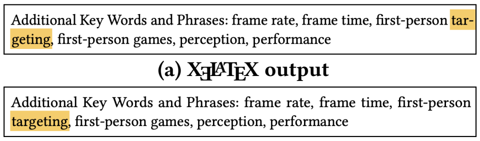
## 3 MOTIVATING STUDY

As a motivating study to inform the design of our experimental methodology, we investigate different methods of comparing documents compiled from different TeX engines to determine how outputs differ between TeX engines. We additionally identify and characterise some typical types of differences observed.

### 3.1 Methodology

We obtained 10 TeX source files from arXiv.org [11], an online open-access archive of scholarly articles, across different taxonomies and document classes (refer to the supplementary materials for a detailed list). Each TeX file was compiled with pdfTeX, XeTeX and LuaTeX, yielding three output PDFs (one from each engine).

We used four different approaches to pairwise comparisons of PDFs. First, we performed pixel-by-pixel comparisons between



Additional Key Words and Phrases: frame rate, frame time, first-person tar-geting, first-person games, perception, performance

(a) XeLaTeX output

Additional Key Words and Phrases: frame rate, frame time, first-person targeting, first-person games, perception, performance

(b) LuaLaTeX output

**Figure 1: Differences in hyphenation in arxiv:2306.01691 [43]**

PDFs produced by different engines. Next, we used text- and font-based comparisons. Lastly, we used features-based comparisons. We converted the first 3 and last 3 pages of each PDF to an image and used the four common computer vision algorithms (namely SIFT, ORB, SSIM, and CWSSIM) in `opencv` to identify similarities between significant structures or regions in each image.

Finally, we studied the differences as indicated by these comparison methods, identified their root causes, and classified them based on their root causes. Both authors then discussed and agreed on the identified root causes to reduce subjectivity and bias in this step.

## 3.2 Findings

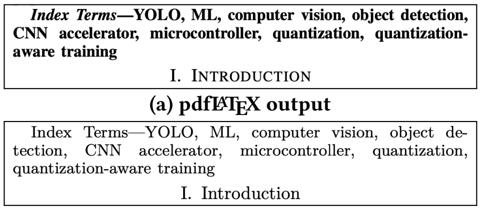
All 10 sources compiled successfully with pdfTeX and LuaTeX. One source failed to compile with XeTeX due to the TeX file importing the `inputenc` package which explicitly throws an error to halt compilation and inform authors that the package is not supported by XeTeX and LuaTeX. This compilation failure was thus expected behaviour and not a bug in any engine. Consequently, we observed that while engines aim to produce relatively similar output, engine-specific LaTeX packages might result in incompatibilities.

Of the 9 sources that successfully compiled a PDF with all three TeX engines, we found differences in all pairwise comparisons between PDFs compiled from the same source on different engines. These differences ranged from inconsistencies in formatting, ligatures (a typographical feature where letters are joined to improve legibility), and content. The supplementary materials provide detailed statistics on these differences. In the rest of this section, we detail the most common types of inconsistencies observed.

*Differences in text spacing.* The most common type of inconsistency we observed was differences in text spacing, which occurred in all 18 pairwise comparisons. This is illustrated in Figure 1, where XeTeX and LuaTeX output the same paragraph of text with a line break at different points due to very small differences in the size of—and space between—individual characters. While this may seem like a small inconsistency, in one document, accumulated differences in text spacing and line breaks resulted in a different number of pages when compiled with pdfTeX compared to XeTeX despite having no differences in content.

*Differences in formatting.* Formatting inconsistencies occurred between pdfTeX and XeTeX for two sources. Figure 2 shows an inconsistency in formatting between the documents produced by pdfTeX (Figure 2a) and XeTeX (Figure 2b), where font styles (specifically **bold text**, *italics* and SMALLCAPS) failed to render in XeTeX.

*Differences in content.* Differences in content occurred between XeTeX and pdfTeX for one source, and between XeTeX and LuaTeX for another source. Figure 3 shows an example of this inconsistency, where XeTeX and pdfTeX rendered different text for a page header. This difference was caused by a hard-coded maximum line height in



***Index Terms*—YOLO, ML, computer vision, object detection, CNN accelerator, microcontroller, quantization, quantization-aware training**

I. INTRODUCTION

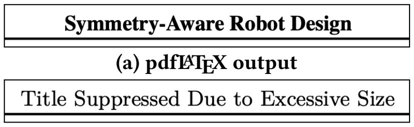
(a) pdfLaTeX output

Index Terms—YOLO, ML, computer vision, object detection, CNN accelerator, microcontroller, quantization, quantization-aware training

I. Introduction

(b) XeLaTeX output

**Figure 2: Differences in font styles in the `IEEEtran` class [55]**



**Symmetry-Aware Robot Design**

(a) pdfLaTeX output

Title Suppressed Due to Excessive Size

(b) XeLaTeX output

**Figure 3: Different page header text for arxiv:2306.00036 [32] using the `icml2023` class (for ICML-2023)**

the `icml2023` document class, which was exceeded in the document compiled by XeTeX (Figure 3b) due to differences in the font used.

## 3.3 Effectiveness of Comparison Methods

We found identifying inconsistencies to be a significant challenge. Due to the complexity of PDF output and the visual basis of determining similarity, we found that comparing PDFs—and identifying different types of inconsistencies between them—posed a key challenge in our initial study. In the rest of this section, we describe how each comparison method performed, and how our findings using each comparison method informed our eventual methodology.

*Pixel-based.* While the naive method of using pixel-based comparisons could detect whether two documents were identical, it failed to differentiate between different types of inconsistencies. Since every document had differences in text spacing, we found this comparison method insufficient as it flagged every comparison as being non-identical but did not help to identify more "interesting" inconsistencies such as missing content or differences in formatting.

*Text-based.* The text-based method of comparison could identify differences in content that were otherwise hard to detect, which helped to prevent false negatives (*i.e.*, overlooked inconsistencies). However, it failed to detect differences in formatting or other visual differences outside of the text content, making it unviable as a standalone comparison method. Additionally, text-based comparisons were prone to false positives (*i.e.*, false alarms due to incorrectly-flagged differences), as visually similar text was often extracted differently due to different handling of font and encoding across different engines. For instance, to overcome the limited number of characters in OT1, pdfTeX renders accented characters by combining a letter with an accent. The "é" character (the letter "e" with an acute diacritic) would be extracted as "´e" in a PDF generated by pdfTeX, and "é" in a PDF generated by LuaTeX or XeTeX, despite being correctly rendered in all three engines.

*Font-based.* Font-based comparison methods are a finer-grained tool to detect formatting inconsistencies. Due to the different font encodings, it was common for documents compiled by different engines to look identical, but use completely different fonts, which resulted in false positives. By comparing the number of fonts instead of performing an exact match, we detected both of the formatting inconsistencies identified in this motivating study.

*Feature-based.* While computer-vision algorithms performed well on images with minimal content, we found that they failed to cope with entire pages of text. Due to the large amount of text, it was computationally infeasible to consider all data present in the image, necessitating trade-offs between the algorithm's accuracy and run time. However, reducing run time by reducing the number of features resulted in high false positive and false negative rates. Feature-based comparisons were thus ineffective in our initial study.

*Summary.* In this initial study, we found that pixel-based comparison methods detected the presence of inconsistencies in all comparisons performed, but were unable to characterise the inconsistencies found, which diminished its effectiveness as it failed to identify "interesting" differences. Text-based comparison methods detected subtle differences in content, but had high false positive rates due to different encoding formats, and could not detect formatting inconsistencies. Font-based comparisons could identify differences in text formatting, but a direct comparison of fonts was not feasible due to large differences in font encodings between TeX engines. Finally, feature-based comparisons were not computationally feasible due to the complexity of the output documents.

Based on these findings, we developed a more refined comparison method using a combination of text-, font-, feature-, and pixel-wise methods to identify inconsistencies in PDF documents. These are further detailed in Section 4.3.

## 4 METHODOLOGY

To answer our research questions, we compared and analysed the compilation outputs of a large number of TeX files. To maximise the efficiency of analysis, we designed and implemented an automated pipeline using a comparative approach to identify inconsistencies.

### 4.1 Dataset

To obtain a large amount of TeX source files, we extracted papers with downloadable source code from arxiv.org. We chose to use arXiv as it presents a high volume and wide variety of papers, most of which have a TeX source. As many of these sources contain multiple files, we identified compilation entry points using regular expressions to simulate a fuzzy search for common file names (*e.g.*, "main.tex", "manuscript.tex") and keywords (*e.g.*, `documentclass`), similar to the approach used by arXiv [14].

For each of the 155 taxonomies in arXiv [3], we queried 3 papers published in June 2023 (arbitrarily chosen) for a target total of 455 papers across a variety of subjects. By covering each taxonomy, we aimed to diversify the testing inputs across different usages of TeX, document classes, and packages. This approach yielded 432 papers in total as some taxonomies had fewer than 3 papers published in that month with TeX sources.

### 4.2 Measuring Interchangeability

To investigate the interchangeability of different TeX engines (**RQ1**) and different versions of TeX Live (**RQ2**), we developed an automated pipeline for compiling TeX sources and comparing output PDFs as shown in Figure 4.

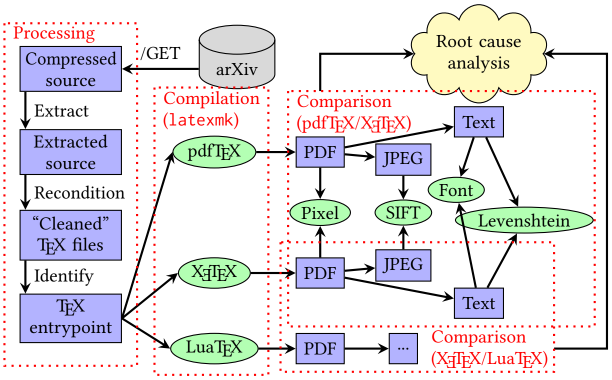


**Figure 4: Overview of the automated pipeline**

We obtained TeX sources by retrieving papers with downloadable source code from arxiv.org. As detailed in the rest of this section, we (1) extracted the compressed sources; (2) cleaned the TeX files by removing patterns that result in expected cases of cross-engine incompatibilities; and (3) identified entry points to compile the TeX sources from. The PDFs compiled by pdfTeX, XeTeX, and LuaTeX were then compared to detect differences across engines.

*Reconditioning.* Each TeX engine provides engine-specific primitives which may cause compilation errors with other TeX engines. Files that use these primitives are thus not engine-agnostic and are expected to produce different output on different TeX engines. Instead of discarding these files, we perform a reconditioning step (as described in Lecoeur et al. [47]) to avoid undefined behaviour and expected compilation failures. By "cleaning" the TeX sources to remove engine-specific primitives such as the `\pdffilesize` primitive that is supported by pdfTeX but not XeTeX or LuaTeX, we aim to reduce the rate of false positives while still maintaining a high volume and diversity of input files.

*Compilation across TeX engines.* To answer **RQ1**, we compiled each file with pdfTeX, XeTeX, and LuaTeX in the 2023 distribution of TeX Live. The `latexmk` program was used to detect and perform the necessary compilation steps instead of invoking the TeX engines directly as some files may require multiple runs or different commands (for instance, due to the use of BibTeX).

*Compilation across TeX Live distributions.* To answer **RQ2**, we compiled each file in the 2020, 2021, 2022, and 2023 release of TeX Live using a minimal `latexmk` configuration. The different distributions of TeX Live were obtained from the official Docker images of historic TeX Live releases [16].

*Comparison.* Pairwise comparisons were performed between PDF files generated from the same input TeX source. For **RQ1**, XeTeX was chosen as the basis of comparison (*i.e.*, we compared XeTeX against pdfTeX, and XeTeX against LuaTeX) because our initial investigation showed that the diffs produced from the pdfTeX/LuaTeX comparison were typically similar to the diffs produced from the pdfTeX/XeTeX comparison. This was because both XeTeX and LuaTeX are Unicode-based, resulting in similar handling of fonts.

For **RQ2**, we performed pairwise comparisons between adjacent versions of TeX Live (2020 versus 2021, 2021 versus 2022, 2022 versus 2023). Additionally, we compared the earliest and latest of these four releases (*i.e.*, 2020 versus 2023) to understand how documents changed over a longer period of time. A sample of inconsistencies was selected to be analysed in **RQ3**, detailed in Section 4.4.

## 4.3 Characterising Differences

Building on the findings from our initial study, we refined the four methods of comparison to identify the different types of inconsistencies across engines, detailed in the rest of this section.

*Pixel-wise comparison.* We performed page-by-page pixel-wise comparisons using the `diff-pdf` CLI tool. This was aimed at detecting small changes in character spacing or other inconsistencies that would be otherwise hard to notice by the human eye.

*Text comparison.* From our initial study, we found that pixel-wise comparisons had limited effectiveness in identifying the kinds of differences found, as the inconsistencies found in pixel-wise comparisons mostly comprised of differences in line breaks and hyphenation, as well as minor differences in text rendering. We used a text-based comparison method as a more fine-grained tool to detect a wider variety of inconsistencies and more subtle differences in text content, as explained in the motivating study.

We first extracted the text from each PDF using the `PyMuPDF` library. To overcome the prevalence of false positives in the motivating study, the extracted text was further processed to ignore differences that would be transparent to users, such as differences in how whitespace is encoded. We then performed pairwise comparisons between processed texts using the Levenshtein edit distance and unique characters present. A non-zero edit distance was used to detect the presence of inconsistencies. Comparing the unique characters present in each PDF, as well as the difference in the number of characters, helped to identify the type of inconsistencies.

*Font-based comparison.* To detect text formatting inconsistencies, we extracted the font information from the text in each PDF using the `PyMuPDF` library. Due to the different font formats and encodings, the exact fonts used differ between engines and we cannot directly compare the fonts used. To overcome this, we also compared the number of fonts present as a proxy for the number of unique styles present in the document, along with font sizes and colours.

*Feature-based comparison.* Since text-based comparison methods cannot identify non-text inconsistencies and a pixel-wise comparison yielded a high rate of false positives in our initial study, we used a feature-based comparison to identify additional inconsistencies.

For each PDF, the first 3 and last 3 pages were converted to JPEG images, and comparison was done using the SIFT algorithm in `opencv`. This subset of pages was chosen to reduce the computational load of the feature extraction step as we found that there was diminishing marginal utility in comparing a larger number of pages, especially for very long documents (*e.g.*, more than 20 pages) where differences in text spacing made page-wise comparisons no longer meaningful. We selected the first and last pages of each document as we observed that these often had the most variation in formatting, with title pages, tables of content, and section titles in the first few pages, as well as references and appendices (including tables and figures) in the final pages.

For each pairwise comparison, a similarity score (from 0 to 1) was calculated to represent how similar the two images are. A score of 0.7 was empirically chosen to be the threshold of considering two images to be significantly inconsistent.

## 4.4 Identifying Root Causes and Triage

For each document identified to have inconsistencies, we manually categorised the types of inconsistency observed. We further analysed the root causes of disproportionately high occurrences of any type of inconsistency. By doing so, we aim to identify common inconsistencies that authors are likely to come across, instead of inconsistencies that could be due to authors' idiosyncrasies.

To answer **RQ3**, we studied the root causes of inconsistencies in a sample of documents. We selected all documents with either compilation failures, inconsistencies introduced only in 2023, or inconsistencies reverted only in 2023. Doing so helped to achieve a diverse sample of inconsistencies, including compile failures, that remain relevant in the current distribution of TEX Live. This accounted for a total of 26 documents (comprising 7 compilation failures, 13 with new inconsistencies, and 6 with reverted inconsistencies). We manually reduced the test case [73], and inspected log files and program outputs, to isolate the cause of inconsistencies; using this approach, we identified unique root causes by verifying the minimal change required to reproduce and eliminate each inconsistency. Finally, we filed bug reports for unexpected behaviours.

*Alignment study.* Since manual classification and analysis is potentially subjective, we used *"negotiated agreement"* [18, 25] to agree on benchmarks and arrive at a consensus.

For identifying and characterising inconsistencies (**RQ1**, **RQ2**), we reviewed the 10 TEX sources from our motivating study, yielding 20 pairwise comparisons. Both authors independently listed and categorised the inconsistencies observed. Using the "negotiated agreement" technique, both authors discussed all divergent classifications and reached a consensus for categorising the rest of the data. Given the lack of existing studies on comparing inconsistent PDFs, this approach seemed reasonable.

A similar approach was taken to mitigate error in identifying the root causes of differences. For all 26 cases studied in **RQ3**, both authors analysed and agreed on the root cause of inconsistencies.

# 5 RESULTS

We found significant inconsistencies in the outputs of different TEX engines and different TEX Live distributions: only 0.2% of documents compiled to identical output with XeLATEX and pdfLATEX, while only 42.1% of produced the same output from TEX Live 2020 to 2023. From further analysing a sample of documents, we identified two new bugs in LATEX packages, and five existing bugs that were fixed independently of this study.

## 5.1 RQ1: TEX Engines Differences

The results obtained for each TEX engine differed significantly. The highest rate of successful compilation was observed with pdfTEX, and it was not uncommon for a paper to successfully compile with pdfTEX but fail to compile on XeTEX or LuaTEX (see Figure 5). 79.2% of documents compiled on all three engines, and 19.4% failed to compile only with XeTEX.

**Figure 5: Compilation success rates for each TeX engine, and number of successful compilations across TeX engines**

| | TeX engine | | |
|---|---|---|---|
| Result/% | pdfTeX | XeTeX | LuaTeX |
| Success | 99.1 | 79.2 | 98.6 |
| Failure | 0.9 | 20.8 | 1.4 |

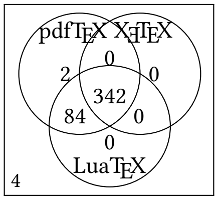


**Table 1: Pairwise comparison results (%)**

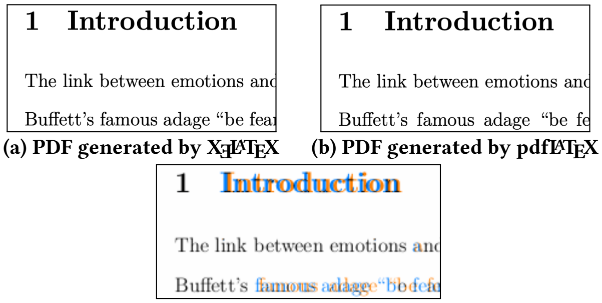
| | Comparing XeLaTeX with: | |
|---|---|---|
| Result/% | pdfLaTeX | LuaLaTeX |
| Compile failure | 20.8 | 20.8 |
| Different output | 78.9 | 77.8 |
| Consistent output | 0.2 | 1.4 |

Compilation failures generally occurred due to incompatibilities between LaTeX packages and TeX engines, not due to TeX engine bugs. We identified the following root causes for compilation failures: (1) a LaTeX package used is incompatible with some TeX engine, or requires action from the author to ensure compatibility (such as including a driver); (2) the LaTeX document class is incompatible with a TeX engine; (3) a syntax is present that relies on a specific TeX engine; (4) and the source file has a syntax error.

From the differences in successful compilation results (*i.e.*, “Different output” in Table 1), we identified several incompatibilities between TeX document classes and TeX engines. Generally, most incompatibilities occurred with XeTeX and LuaTeX (*i.e.*, many packages were compatible only with pdfTeX). The common types of differences caused by different document classes were:

- **Text spacing**, referring to the spaces or misalignments between individual characters (see Figure 6);
- **Line breaks**, referring to how text (*e.g.*, a paragraph) is divided into multiple lines (see Figure 1);
- **Number of pages** in the output PDF;
- **Missing content**, where one output PDF contains content that another does not (see Figure 3);
- **Inconsistent styles**, such as italics or bold text (Figure 2), differences in ligatures, or differences in how a character is rendered (Figure 7);
- **References**, where citations or bibliographies are rendered differently (see Figure 8); and
- **Images**, such as the size and position of images.

The occurrences of these differences are summarised in Table 2. From analysing document classes with a disproportionately high rate of inconsistencies, we identified several popular document classes that are compatible with pdfTeX but incompatible with other engines. For instance, due to differences in how each engine handles font loading, the `IEEEtran` class is compatible with pdfTeX only; all styles (such as italics, bold, and small-caps) are not rendered in other engines (Figure 2). `IEEEtran` is the template recommended by IEEE (Institute of Electrical and Electronics Engineers) for use in conference proceedings [8]. Similar incompatibilities were observed, though to a lesser extent, with the `mnras` class (for the Monthly Notices of the Royal Astronomical Society [6]).


1 Introduction

The link between emotions anc

Buffett’s famous adage “be fear

**(a) PDF generated by XeLaTeX**

1 Introduction

The link between emotions anc

Buffett’s famous adage “be fe

**(b) PDF generated by pdfLaTeX**

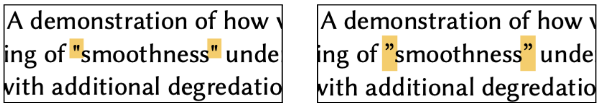
1 Introduction

The link between emotions anc

Buffett’s famous adage “be fe

**(c) Overlaying PDFs to highlight differences (🟦 XeLaTeX; 🟧 pdfTeX)**

**Figure 6: Different text spacing (arXiv:2306.12602 [69])**

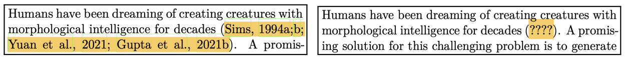

A demonstration of how v
ing of "smoothness" unde
vith additional degradatio

**(a) PDF generated by pdfLaTeX**

A demonstration of how v
ing of ”smoothness” unde
vith additional degradatio

**(b) PDF generated by XeLaTeX**

**Figure 7: Different quotation marks (arXiv:2306.01691 [43])**


Humans have been dreaming of creating creatures with morphological intelligence for decades (Sims, 1994a;b; Yuan et al., 2021; Gupta et al., 2021b). A promis-

**(a) PDF generated by XeLaTeX**

Humans have been dreaming of creating creatures with morphological intelligence for decades (????). A promising solution for this challenging problem is to generate

**(b) PDF generated by LuaLaTeX**

**Figure 8: Differences in references (arXiv:2306.00036 [32])**

The Venn diagrams in Figure 9 illustrate how the different inconsistencies are distributed when comparing pdfTeX and XeTeX. We observed that relatively higher numbers (15 or greater) of documents containing a given set of inconsistencies tended to occur more often compared to smaller numbers (3 or below). This is likely because inconsistencies tend to occur based on the document class of the TeX file, and thus input sources using the same document class would have the same types of inconsistencies. Small numbers in the Venn diagram were likely due to documents where the author imported a specific LaTeX package that was incompatible with some TeX engine, or due to the author using engine-specific syntax that triggered a specific type of inconsistency not usually observed with the document class.

## 5.2 RQ2: TeX Live Version Differences

*Overview.* We found significant differences in the compilation results of different versions of TeX Live. Similar to the inter-engine comparison, the most common type of inconsistencies were differences in text spacing, which were usually relatively minor: such differences were mostly small shifts in the position of characters on a page, and did not affect the information conveyed in the document. However, more “major” differences affecting the content of the document were still prevalent.

Table 3 shows the pairwise comparison results across four years from 2020 to 2023 (the supplementary materials provide more detailed statistics on the inconsistencies in each year). Notably, TeX Live 2021 to 2023 was mostly stable, with over 80% of documents compiling to the same output across all three years. However, the shift from TeX Live 2020 to TeX Live 2021 resulted in significant discrepancies across most documents, where only 20.4% of documents produced the same output across both versions of TeX Live.

**Table 2: Differences across common document classes (XeLaTeX vs pdfLaTeX)**

| | % of papers with difference (XeLaTeX versus pdfLaTeX) | | | | | | |
|---|---|---|---|---|---|---|---|
| Class (count) | Missing styles | Missing content | Number of pages | Images | Text spacing | Line breaks | References |
| All compiled (342) | 21.9 | 2.3 | 17.8 | 27.8 | 98.3 | 96.2 | 0.9 |
| amsart (65) | 6.2 | 0.0 | 21.5 | 21.5 | 96.8 | 88.9 | 0.0 |
| revtex4-2 (43) | 7.0 | 0.0 | 25.6 | 37.2 | 95.3 | 95.3 | 2.3 |
| IEEEtran (27) | 63.0 | 0.0 | 63.0 | 70.4 | 100.0 | 88.9 | 0.0 |
| elsarticle (26) | 19.2 | 0.0 | 30.8 | 50.0 | 100.0 | 100.0 | 0.0 |
| revtex4-1 (23) | 0.0 | 0.0 | 30.4 | 52.2 | 100.0 | 100.0 | 0.0 |
| acmart (21) | 19.0 | 0.0 | 28.6 | 52.4 | 95.0 | 95.0 | 0.0 |

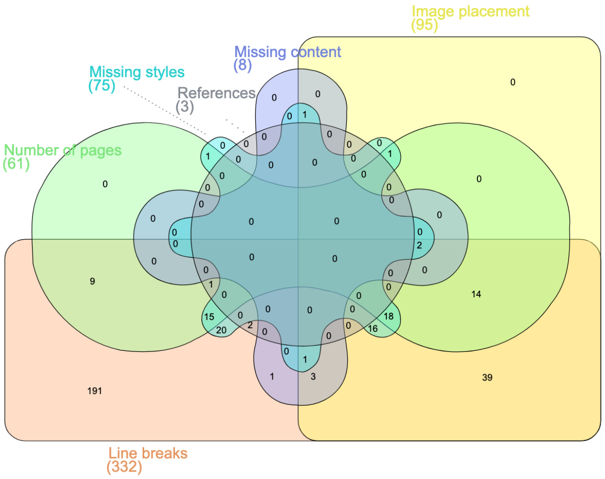


**Figure 9: Distribution of inconsistencies observed between pdfTeX and XeTeX**

**Table 3: Comparison results (%) from TeX Live 2020 to 2023**

| | TeX Live versions compared | | | |
|---|---|---|---|---|
| Comparison result /% | '20/'21 | '21/'22 | '22/'23 | '20/'23 |
| Identical PDFs | 20.4 | 80.1 | 85.2 | 30.3 |
| Different PDFs | 69.4 | 19.2 | 13.9 | 68.1 |
| Compile failure | 1.2 | 0.7 | 0.9 | 1.6 |

*Breaking changes in compilation flags.* From 2020 to 2021, a significant proportion (48.6%) of documents had discrepancies in references. This was due to a change in the default value of the `bibtex_fudge` option used by `latexmk`, an automated script included in the TeX Live distribution for compiling LaTeX documents. This regression affected the compilation of documents with BibTeX citations, and was patched in the subsequent version of `latexmk`. After accounting for the change in compilation commands required in TeX Live 2020, the proportion of documents with discrepancies in references fell from 48.6% to only 2.8%.

*Types of inconsistencies observed.* After accounting for changes in compilation commands, the amount of inconsistencies found decreased across all categories but still remained significant, with only 54.5% of documents compiling to the same output across both years. Differences in text spacing were the most common discrepancy observed, affecting 47.2% of documents; such differences were particularly common in document classes like `acmart` (76%), `amsart` (51%) and `mnras` (88%), as well as the packages for `neurips` (90%) and `icml2023` (86%). Differences in text spacing throughout a document may result in more major inconsistencies: of the eight documents using the `mnras` class, one document compiled to a different number of pages in 2020 and 2021 due to accumulated differences in text spacing despite having no difference in text content.

Similar to the cross-engine comparison, other discrepancies were observed as well, ranging from differences in font styles to missing content and missing references. Table 4 shows the distribution of the discrepancies observed across the years. Notably, from 2021 to 2022, we observed a relatively higher proportion of inconsistencies related to references and text content.

*Missing references.* We observed a relatively higher incidence of inconsistencies with references from 2021 to 2022 (affecting 10.0% of documents, compared to under 2.8% from 2020 to 2021 and 2.3% from 2022 to 2023). This was largely due to a difference in `latexmk`'s automated compilation in response to syntax errors related to bibliographies. In the 2021 TeX Live distribution, `latexmk` would often continue to run compilation steps after encountering a BibTeX error, allowing some references to be rendered in spite of syntax errors; whereas in the 2022 distribution, `latexmk` would often stop attempting compilation, resulting in differences in the references across both years. This change in behaviour was likely not a bug as it resulted from authors' syntax errors being handled differently, and errors were appropriately logged to warn authors.

*Changes in output over four years.* Table 5 shows the changes in compilation output over time, with "○" denoting identical PDF output between two years and "●" denoting different output. Even after accounting for differences in compilation flags, only 42.1% of documents had consistent output across all four versions of TeX Live. Of the documents that did not compile to the same output in all four years, the majority had differences in output only between the 2020 and 2021 versions (accounting for 27.1% of the total). The relatively high rate of inconsistencies from 2020 to 2021 is largely attributed to differences in text spacing as described above. Interestingly, Table 5 shows how some changes were reverted in later distributions of TeX Live, with 1.7% of documents having consistent output when compiled with the 2020 and 2023 versions, but not intermediate versions. We further discuss these cases in Section 5.3.

## 5.3 RQ3: Root Cause Analysis

Overall, we analysed 26 documents (the specific documents selected are listed in the supplementary materials) and identified 10 unique

**Table 4: Differences across different TEX Live versions (2020–2023)**

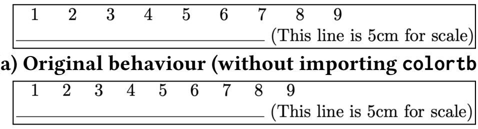
| | % of papers with difference (comparing TEX Live versions) | | | | | | |
|---|---|---|---|---|---|---|---|
| Versions | Missing styles | Missing content | Number of pages | Images | Text spacing | Line breaks | References |
| '20/'23 | 6.0 | 39.4 | 34.0 | 0.7 | 65.7 | 44.7 | 44.4 |
| '20/'23 (∗) | 5.8 | 3.5 | 7.9 | 1.6 | 51.4 | 14.8 | 11.6 |
| '20/'21 | 5.6 | 45.4 | 38.4 | 0.0 | 67.1 | 49.8 | 48.6 |
| '20/'21 (∗) | 4.6 | 2.3 | 2.8 | 0.7 | 42.4 | 7.6 | 2.8 |
| '21/'22 | 1.4 | 6.5 | 5.3 | 0.0 | 15.5 | 8.8 | 10.0 |
| '22/'23 | 1.6 | 0.7 | 0.5 | 0.7 | 13.0 | 3.2 | 2.3 |

(∗) Compiled with extra compilation flags due to changes in `latexmk` default options

**Table 5: Comparison results over time (2020 to 2023)**

○ identical output; ● different output

| Comparison results | | | | Count |
|---|---|---|---|---|
| '20/'21 | '21/'22 | '22/'23 | '20/'23 | |
| ○ | ○ | ○ | ○ | 42.1% |
| ● | ○ | ○ | ● | 27.1% |
| ● | ● | ○ | ● | 8.1% |
| ○ | ● | ○ | ● | 7.9% |
| ● | ○ | ● | ● | 6.7% |
| ○ | ○ | ● | ● | 3.0% |
| ● | ● | ● | ● | 2.8% |
| ● | ○ | ● | ○ | 1.2% |
| ○ | ● | ● | ● | 0.7% |
| ○ | ● | ● | ○ | 0.5% |



**(a) Original behaviour (without importing `colortbl`)**

**(b) Importing `colortbl`**

**Figure 10: Decreased spacing when `colortbl` is imported**

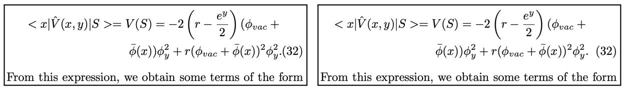
root causes for the inconsistencies found. Of the 10 root causes, two were new bugs identified by this study; five were bugs fixed independently of this study; three were expected behaviour.

*New bugs.* In first bug we identified, we found that importing the `colortbl` package decreased the horizontal spacing within tables (shown in Figure 10), even without using any commands or functionality provided by the package. The developers responded to our bug report and confirmed that this behaviour is a bug due to inconsistent behaviour of the core LATEX `array` package [27, 54].

The second bug was with the `jmlr` document class: using this class resulted in a compilation failure in 2023 as it had not been updated to work with the TEX Live 2023 distribution. We filed a bug report that has yet to be triaged [12].

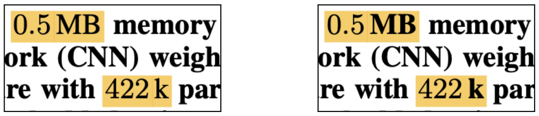
*Existing bugs.* Our analysis identified five existing (*i.e.*, identified or fixed independently of this study) bugs. The first bug was in the core LATEX package, where inserting newlines in footnotes using the "\\" macro would fail compilation despite having valid syntax. This bug was reported and fixed in TEX Live 2021 [35].

The second bug was caused by the `hyperref` package, where importing the package resulted in increased spacing between an equation environment and a theorem environment. However, the change in spacing was desirable (*i.e.*, the behaviour in 2023 was "more correct"), as it was caused by fixing an existing bug rather than introducing a new bug.


From this expression, we obtain some terms of the form
From this expression, we obtain some terms of the form

**(a) Output from TEX Live 2020** **(b) Output from TEX Live 2021**

**Figure 11: Math equations rendered slightly leftwards in TEX Live 2021, compared to TEX Live 2020 (arXiv:2306.03237 [19])**



**(a) Output from TEX Live 2020** **(b) Output from TEX Live 2021**

**Figure 12: SI units (highlighted) are bold in TEX Live 2021, but not TEX Live 2020 (arXiv:2306.00001 [55])**

The third bug was in the `revtex4-2` document class. We observed three documents from the `revtex4-2` document class compiled to the same output in TEX Live 2020 and 2023, but TEX Live 2021 compiled with slightly misaligned equations in the `eqnarray` environment (Figure 11), which was the environment recommended in the REVTEX 4.2 Author's Guide [64]. This change was due to a bug in the *"Detection of \eqnarray in Newer LaTeX"* which was fixed in the 2023 distribution [9]. The fourth bug was similar, but occurred in the core LATEX package with the behaviour of the built-in `\eqno` macro in LATEX when handling newlines, resulting in different indentation of the paragraph after an equation for two documents. This change was introduced in the 2022 distribution and recognised as a bug that was fixed in the 2023 distribution [5].

Finally, the fifth bug was caused by the `siunitx` package. In TEX Live 2021 and 2022, font styles were inconsistently applied to quantities and units as shown in Figure 12. This bug was fixed in the 2023 distribution.

*Expected behaviour.* Three root causes of inconsistencies, accounting for 16 documents, were expected behaviour. Three documents failed to compile in all years due to syntax errors made by authors. Two documents failed to compile in 2020 as they imported the `tabularray` package, which was introduced only in 2021.

In the remaining 11 documents, the use of the tilde ("~") alignment character resulted in differences in the spacing and alignment of equations from TEX Live 2022 to 2023. The developers deemed this intended behaviour as the *"current behavior ... is more correct"*, and indicated that the change was due to a correction in how the tilde character was defined [26]. While the change was listed in the release notes as *"Aligning status of tilde with other active characters"* [7], the impact on authors was not directly documented.

## 6 THREATS TO VALIDITY

*Bias towards pdfTEX.* As seen in Figure 5, pdfTEX had the highest successful compilation rate by far. Given that pdfTEX is the oldest and most popular TEX engine amongst the three, this result was unsurprising. The high compilation rate with pdfTEX may also have been because arXiv "fully supports and automatically recognizes pdfLATEX" but not XELATEX or LuaLATEX [13], and thus the TEX sources retrieved from arXiv are more likely to be biased towards pdfLATEX.

*Identifying and characterising differences.* When a large number of inconsistencies were present in a document, we might have missed some inconsistencies. For instance, when a document compiled with differences in font styles, text spacing, references, and text content across different engines, it may be difficult to identify the root cause of one specific inconsistency due to the multitude of confounding factors. We mitigated this by using a large number and variety of papers, as a diverse input sample might reveal different combinations of inconsistencies and thus allow us to better identify their root causes. One example of this would be the `\eqnarray` bug in REVTeX (see Section 5.2). This bug would have affected all 43 documents using the `revtex4-2` document class. In three of these documents, this bug was the only inconsistency observed, allowing it to be more easily detected and diagnosed. In the remaining 40 documents, the bug occurred alongside other inconsistencies, making it harder to both detect and characterise. By using a large number of input files, we aim to improve the diversity of our results to obtain both isolated instances of inconsistencies, as well as cases where different inconsistencies may occur in tandem.

## 7 IMPLICATIONS

In this section, we evaluate the actionable insights of our study, and discuss how this relates to authors of TEX documents and developers in the TEX ecosystem.

*Restricted choice of engines.* We found that several popular document classes associated with academic journals were incompatible with XETEX and LuaTEX. Thus, if an author wishes to use one of these document classes in their TEX document, they would not be able to use these TEX engines even if they took care to ensure that their document was engine-agnostic. This is because the document class itself contained engine-specific behaviour or dependencies, and the author of a TEX document using such a class would have to modify the document class itself, which is often not allowed in contexts such as conference submissions. This is a problem for authors who cannot use pdfLATEX (for instance, pdfLATEX has poor support for bidirectional text, making XELATEX or LuaLATEX a more popular choice for languages like Arabic [42]). Thus, while engines are purported to be *"a drop-in replacement"* [1] most of the time, inconsistencies between engines are common due to the prevalence of document classes that do not support cross-engine compatibility. Authors should thus test their documents to ensure that compilation works as intended, especially if they intend to use engines outside of pdfTEX, to avoid negative outcomes (such as desk-rejection) that result from unintended output.

*Cascading effects of incompatibilities.* We found several incompatibilities between LATEX packages and TEX engines. Such incompatibilities have a significant impact on the TEX ecosystem as a whole: if a LATEX package is incompatible with a particular TEX engine, then a document class that uses this package will likely be incompatible with the engine as well. Such incompatibilities further limit authors' choices of TEX engines, as they may be limited to choosing TEX engines that are compatible with a given document class that they intend to use. A reduced user base may have cascading effects on the development of a TEX engine, as exposure to a community has been shown to incentivise contributions to open-source software [59].

*Effects of version upgrades.* Over different distributions of TEX Live, we observed differences in the behaviour of compilation options (such as the usage of `latexmk`), LATEX packages, and document classes (including those used by popular conferences). These discrepancies suggest that it is important for authors of TEX documents to ensure that their versions of TEX Live are up-to-date, especially if they are required to adhere to a specific LATEX format or document specification. For instance, an author preparing a document using an older version of TEX Live may fail to adhere to a conference's page limit when the document is compiled with the latest version, due to differences in text spacing and line breaks which result in a different number of pages. Such differences may occur even if there have been no changes in the document class. Authors can mitigate these risks by ensuring that their TEX Live distributions are updated or aligned with the versions recommended by their submission guidelines.

*Managing version updates.* In spite of the importance of keeping updated to an appropriate version of TEX Live, the prevalence of inter-version inconsistencies might deter authors from doing so, especially if they are unable to determine how their documents or workflows may change. While the LATEX project maintains a comprehensive changelog and release newsletter to document changes, the practical impact of changes on authors is not always directly explained. For instance, a change in how the tilde character was internally defined was listed in the release notes as *"Aligning status of tilde with other active characters"* [7], which does not mention potential changes in how text might be rendered around the tilde character. Due to the complexity of a typesetting system, it can be difficult to document how authors may be impacted, and changelogs may not be sufficiently informative for authors less familiar with TEX internals. Furthermore, the impact of changes on authors is difficult to predict as changes in "core" functionality, such as the LATEX package itself, will inevitably affect the downstream packages depending on it. Due to the complexity of the TEX ecosystem, it is virtually impossible for authors to predict how their documents may be affected by version changes. Authors may mitigate the risk of unexpected or undesired inconsistencies by manually vetting their compiled documents, as it is difficult to determine how their documents will change otherwise.

*Handling warnings and errors.* Several discrepancies across TEX Live versions were due to different ways of handling warnings and errors. For example, in the 2021 distribution, `latexmk` would continue to attempt compilation on documents with syntax errors in their bibliography file; whereas in the 2022 distribution, `latexmk` would abort compilation. While such differences might fall under undefined behaviour as they result from authors' syntax errors, acting on any warnings or errors output during compilation can help authors to avoid unexpected differences across TEX Live versions.

*Finding bugs.* Our approach identified various bugs in the TEX ecosystem through inter-version comparisons, as well as significant inconsistencies in cross-engine comparisons. This pipeline may thus function as an automated differential testing pipeline for developing TEX-related software by providing a varied test corpus, allowing developers to test how their software may interact with other software in the TEX ecosystem. Due to the complexity of the TEX ecosystem, where authors tend to use a large variety of packages and document classes which may be updated independently of each other, such an approach would be highly beneficial in uncovering edge cases as it uncovers issues that might not be found in isolated tests. This helps developers test how their software might be affected by, or result in, changes in the TEX ecosystem over a large number of diverse use cases. Besides TEX-related differential testing campaigns, our analysis of the efficacy of various PDF comparison methods could inform the design of other testing approaches that similarly focus on human-readable documents.

## 8 RELATED WORK

To our knowledge, there is no existing work on testing TEX engines or comparing TEX Live distributions. In this section, we summarise related works that discuss similar approaches or adjacent goals.

*Studying the TEX ecosystem.* Much of TEX-related published work focuses on improving author experience through LATEX packages [22, 36, 52, 56], automatically generating LATEX code [41, 58, 61, 68, 70], or developing TEX-adjacent software like syntax fixers [75] and command-line tools [20, 21]. We found relatively less literature studying the TEX ecosystem itself. The closest related work is a study on vulnerabilities in the TEX ecosystem [46], and a study on the role of LATEX, BibTEX, and metadata in publishing articles [23].

*Comparing PDF documents.* Comparing, and verifying the correctness of, PDF documents was a key challenge in this study. An existing attempt at an automated PDF format checker was effective for all PDFs except those generated by LATEX, as the underlying PDF parser could not retrieve sufficient font information in LATEX-generated documents [71]. Another study compared PDF documents using image similarity algorithms to effectively detect inconsistencies [45]. However, this approach was not effective in our study as we had a higher threshold for permissible variation between documents, resulting in excessive false positives.

*Identifying visual discrepancies.* One difficulty in this study was identifying output differences that were visible and significant. Donaldson et al. [31] used the chi-squared distance algorithm to compare image histograms and identify meaningful differences when testing graphics shader compilers, as slightly different results are allowed due to compiler optimisations. We found similar methods unsuitable as differences in line breaks and hyphenation resulted in noise that obscured other types of inconsistencies. Other methods of quantifying image similarity include summing the absolute distance between components of vectors [63], and pixel-wise algorithms like Structural Similarity Index (SSI) and Mean Square Error (MSE) [51]. These algorithms were unsuitable in our context as the different TEX engines often generated pixel-wise differences that were visually unnoticeable, resulting in false positives.

*Differential testing.* Differential testing is a form of random testing that identifies a test case as a candidate to cause a bug if it results in different outputs in two (or more) comparable systems [53]. It has been used to find bugs in various domains, such as document processors [48], software libraries [29, 38], program analysers [44, 62], database systems [50, 65, 74], runtime engines [28, 40, 57], and most notably compilers [34, 49, 67, 72]. Given the similarity in methodology, our study can be seen as a form of differential testing. However, different from these works, our outputs are neither well-defined nor not expected to be perfectly identical. It was thus challenging to identify whether a discrepancy was expected or not.

*Sanitising.* Sanitisers are error-detection frameworks that check for various potential bugs, such as integer overflows [30] or out-of-bound memory accesses [60, 66]. Studies on differential testing on compilers have attempted to detect and reduce undefined behaviour in test cases using compiler sanitisers. For instance, the GrayC tool generates C programs and filters out invalid programs using Frama-C and Clang/LLVM compiler sanitisers [34]. While sanitising may have reduced the amount of false positives in our paper, we found sanitising to be impractical as we could not account for undefined behaviour in the individual LATEX packages used by each test case.

## 9 CONCLUSION

We have developed an automated pipeline for identifying inconsistencies between TEX engines and TEX Live distributions. Using this approach, we found a significant extent of inconsistencies throughout the TEX ecosystem, ranging from changes in default compilation flags and formatting in document classes to behaviours of LATEX packages and the core LATEX package itself. We found two new bugs in LATEX packages, along with five existing bugs and other obscured differences. We characterised the inconsistencies found to demonstrate the typical ways in which TEX engines may differ beyond their explicitly listed differences, as well as what authors can expect from different versions of TEX Live. By focusing on the TEX ecosystem as a whole, we offer actionable and practical insights for authors of TEX documents to manage the constantly-evolving software and avoid undesirable outcomes. We believe this study will also be beneficial for developers by empirically demonstrating the relationship between different parts of the ecosystem, how inconsistencies manifest, and how it may affect their users. Finally, our automated pipeline can be additionally used as a differential testing approach to gather a diverse test corpus and uncover more bugs and edge cases in the TEX ecosystem.

## 10 DATA AVAILABILITY

Our automated tool is available on GitHub at https://github.com/ [redacted]. This software, along with our experimental results and configurations for reproducibility, is archived for long-term storage at https://zenodo.org/records/10778054.

# REFERENCES

[1] TeXFAQ 2018. *What are XeTeX and LuaTeX?* TeXFAQ. https://texfaq.org/FAQ-xetex-luatex

[2] Overleaf 2022. *Starting a New Era with 11 Million Users and 100 Million Projects.* Overleaf. https://www.overleaf.com/blog/starting-a-new-era-with-11-million-users-and-100-million-projects

[3] arXiv 2023. *Category Taxonomy.* arXiv. https://arxiv.org/category_taxonomy

[4] TeX Users Group 2023. *History of TeX.* TeX Users Group. https://www.tug.org/whatis.html

[5] latex3 2023. *ignoring space after \eqno #1059.* latex3. https://github.com/latex3/latex2e/issues/1059

[6] The Royal Astronomical Society 2023. *Instructions to Authors.* The Royal Astronomical Society. https://academic.oup.com/mnras/pages/general_instructions

[7] The LaTeX Project 2023. *LaTeX2e News Issue 38.* The LaTeX Project. https://www.latex-project.org/news/latex2e-news/ltnews38.pdf

[8] Institute of Electrical and Electronics Engineers 2023. *Manuscript Templates for Conference Proceedings.* Institute of Electrical and Electronics Engineers. https://www.ieee.org/conferences/publishing/templates.html

[9] American Physical Society 2023. *REVTeX Home Page.* American Physical Society. https://journals.aps.org/revtex

[10] TeX Users Group 2023. *What are TEX and its friends?* TeX Users Group. https://www.ctan.org/tex

[11] Cornell University 2024. *arXiv.org e-Print archive.* Cornell University. https://arxiv.org/

[12] 2024. Setting curr file breaks compilation with images. https://www.dickimaw-books.com/bugtracker.php?key=273.

[13] arXiv 2024. *TeX Submissions.* arXiv. https://info.arxiv.org/help/submit_tex.html

[14] arXiv 2024. *TeX::AutoTeX - automated processing of (La-)TeX sources - metacpan.org.* arXiv. https://metacpan.org/pod/TeX::AutoTeX#SYNOPSIS

[15] TeX Users Group 2024. *TeXLive Availability.* TeX Users Group. https://www.tug.org/texlive/acquire.html

[16] TeXlive 2024. *texlive/texlive - Docker Image | Docker Hub.* TeXlive. https://hub.docker.com/r/texlive/texlive

[17] TeX Users Group 2024. *Welcome to TeX Live.* TeX Users Group. https://www.tug.org/texlive//Contents/live/readme-html.dir/readme.en.html

[18] Rabe Abdalkareem, Olivier Nourry, Sultan Wehaibi, Suhaib Mujahid, and Emad Shihab. 2017. Why do developers use trivial packages? an empirical case study on npm. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (Paderborn, Germany) *(ESEC/FSE 2017)*. Association for Computing Machinery, New York, NY, USA, 385–395. https://doi.org/10.1145/3106237.3106267

[19] Ivan Arraut. 2023. Gauge symmetries and the Higgs mechanism in Quantum Finance. *Europhysics Letters* 143, 4 (Aug. 2023), 42001. https://doi.org/10.1209/0295-5075/acedce

[20] Haim Bar and HaiYing Wang. 2021. Reproducible Science with LATEX. *Journal of Data Science* (2021), 111–125. https://doi.org/10.6339/21-jds998

[21] Paschalis Bizopoulos. 2023. A Makefile for Developing Containerized LaTeX Technical Documents. arXiv:2005.12660 [cs.SE]

[22] Christof Bless, Ildar Baimuratov, and Oliver Karras. 2023. SciKGTeX – A LaTeX Package to Semantically Annotate Contributions in Scientific Publications. arXiv:2304.05327 [cs.DL]

[23] Joppe W. Bos and Kevin S. McCurley. 2023. LaTeX, metadata, and publishing workflows. arXiv:2301.08277 [cs.DL]

[24] François Brischoux and Pierre Legagneux. 2009. Don't Format Manuscripts. *Scientist (Philadelphia, Pa.)* 23 (04 2009), 24–24.

[25] John L. Campbell, Charles Quincy, Jordan Osserman, and Ove K. Pedersen. 2013. Coding In-depth Semistructured Interviews: Problems of Unitization and Intercoder Reliability and Agreement. *Sociological Methods & Research* 42, 3 (2013), 294–320. https://doi.org/10.1177/0049124113500475 arXiv:https://doi.org/10.1177/0049124113500475

[26] [redacted]. 2024. Changes in horizontal spacing in some equations. https://github.com/latex3/latex2e/[redacted].

[27] [redacted]. 2024. Importing colortbl changes horizontal spacing. https://github.com/davidcarlisle/dpctex/[redacted].

[28] Yuting Chen, Ting Su, Chengnian Sun, Zhendong Su, and Jianjun Zhao. 2016. Coverage-directed differential testing of JVM implementations. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Santa Barbara, CA, USA) *(PLDI '16)*. Association for Computing Machinery, New York, NY, USA, 85–99. https://doi.org/10.1145/2908080.2908095

[29] Yuting Chen and Zhendong Su. 2015. Guided differential testing of certificate validation in SSL/TLS implementations. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015)*. Association for Computing Machinery, New York, NY, USA, 793–804. https://doi.org/10.1145/2786805.2786835

[30] Will Dietz, Peng Li, John Regehr, and Vikram Adve. 2015. Understanding Integer Overflow in C/C++. *ACM Trans. Softw. Eng. Methodol.* 25, 1, Article 2 (dec 2015), 29 pages. https://doi.org/10.1145/2743019

[31] Alastair F. Donaldson, Hugues Evrard, Andrei Lascu, and Paul Thomson. 2017. Automated Testing of Graphics Shader Compilers. *Proc. ACM Program. Lang.* 1, OOPSLA, Article 93 (oct 2017), 29 pages. https://doi.org/10.1145/3133917

[32] Heng Dong, Junyu Zhang, Tonghan Wang, and Chongjie Zhang. 2023. Symmetry-Aware Robot Design with Structured Subgroups. arXiv:2306.00036 [cs.AI]

[33] Hàn Thê ́ Thành et al. 2023. *The pdfTeX user manual.* Free Software Foundation. https://texdoc.org/serve/pdftex-a.pdf/0

[34] Karine Even-Mendoza, Arindam Sharma, Alastair F. Donaldson, and Cristian Cadar. 2023. GrayC: Greybox Fuzzing of Compilers and Analysers for C. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis* (Seattle, WA, USA) *(ISSTA 2023)*. Association for Computing Machinery, New York, NY, USA, 1219–1231. https://doi.org/10.1145/3597926.3598130

[35] FrankMittelbach. 2021. \\variants in tables are not robust. https://github.com/latex3/latex2e/issues/548.

[36] Pablo García Risueño, Apostolos Syropoulos, and Natàlia Vergés. 2016. Ideograms for Physics and Chemistry. *Foundations of Physics* 46, 12 (Nov. 2016), 1713–1721. https://doi.org/10.1007/s10701-016-0044-5

[37] Alex Gaudeul. 2007. Do Open Source Developers Respond to Competition? The LATEX Case Study. *Review of Network Economics* 6, 2 (2007). https://doi.org/doi:10.2202/1446-9022.1119

[38] Patrice Godefroid, Daniel Lehmann, and Marina Polishchuk. 2020. Differential regression testing for REST APIs. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Virtual Event, USA) *(ISSTA 2020)*. Association for Computing Machinery, New York, NY, USA, 312–323. https://doi.org/10.1145/3395363.3397374

[39] Hans Hagen, Harmut Henkel, Taco Hoekwater, and Luigi Scarso. 2023. *LuaTeX Reference Manual.* LuaTeX development team. http://mirrors.ibiblio.org/CTAN/systems/doc/luatex/luatex.pdf

[40] Shuyao Jiang, Ruiying Zeng, Zihao Rao, Jiazhen Gu, Yangfan Zhou, and Michael R. Lyu. 2023. Revealing Performance Issues in Server-side WebAssembly Runtimes via Differential Testing. arXiv:2309.12167 [cs.SE]

[41] Pratik Kayal, Mrinal Anand, Harsh Desai, and Mayank Singh. 2022. Tables to LaTeX: structure and content extraction from scientific tables. *International Journal on Document Analysis and Recognition (IJDAR)* 26, 2 (Oct. 2022), 121–130. https://doi.org/10.1007/s10032-022-00420-9

[42] Jonathan Kew. 2005. XeTeX, the Multilingual Lion: TEX meets Unicode and smart font technologies. In *TUG 2005 Proceedings* (Wuhan, China) *(2005 International Typesetting Conference)*. TUGBoat, Portland, Oregon, U.S.A, 115–124.

[43] Devi Klein, Josef Spjut, Ben Boudaoud, and Joohwan Kim. 2023. The Influence of Variable Frame Timing on First-Person Gaming. arXiv:2306.01691 [cs.GR]

[44] Christian Klinger, Maria Christakis, and Valentin Wüstholz. 2018. Differentially Testing Soundness and Precision of Program Analyzers. arXiv:1812.05033 [cs.SE]

[45] Tomasz Kuchta, Thibaud Lutellier, Edmund Wong, Lin Tan, and Cristian Cadar. 2018. On the correctness of electronic documents: studying, finding, and localizing inconsistency bugs in PDF readers and files. *Empir. Softw. Eng.* 23, 6 (Dec. 2018), 3187–3220.

[46] Guilhem Lacombe, Kseniia Masalygina, Anass Tahiri, Carole Adam, and Cédric Lauradoux. 2021. Can You Accept LaTeX Files from Strangers? Ten Years Later. arXiv:2102.00856 [cs.CR]

[47] Bastien Lecoeur, Hasan Mohsin, and Alastair F. Donaldson. 2023. Program Reconditioning: Avoiding Undefined Behaviour When Finding and Reducing Compiler Bugs. *Proc. ACM Program. Lang.* 7, PLDI, Article 180 (jun 2023), 25 pages. https://doi.org/10.1145/3591294

[48] Shuxin Li and Manuel Rigger. 2024. Finding XPath Bugs in XML Document Processors via Differential Testing. arXiv:2401.05112 [cs.SE]

[49] Christopher Lidbury, Andrei Lascu, Nathan Chong, and Alastair F. Donaldson. 2015. Many-core compiler fuzzing. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Portland, OR, USA) *(PLDI '15)*. Association for Computing Machinery, New York, NY, USA, 65–76. https://doi.org/10.1145/2737924.2737986

[50] Wei Lin, Ziyue Hua, Luyao Ren, Zongyang Li, Lu Zhang, and Tao Xie. 2022. GDsmith: Detecting Bugs in Graph Database Engines. arXiv:2206.08530 [cs.DB]

[51] T. Malathi and Manas Bhuyan. 2017. Performance analysis of Gabor wavelet for extracting most informative and efficient features. *Multimedia Tools and Applications* 76 (03 2017). https://doi.org/10.1007/s11042-016-3414-2

[52] Luke Marris, Ian Gemp, Siqi Liu, Joel Z. Leibo, and Georgios Piliouras. 2024. Visualizing 2x2 Normal-Form Games: twoxtwogame LaTeX Package. arXiv:2402.16985 [cs.GT]

[53] William M. McKeeman. 1998. Differential Testing for Software. *Digit. Tech. J.* 10, 1 (1998), 100–107. https://www.hpl.hp.com/hpjournal/dtj/vol10num1/vol10num1art9.pdf

[54] Frank Mittelbach. 2024. array package exhibits a surprising spacing difference when math cells end in explicit spaces. https://github.com/latex3/latex2e/[redacted].

[55] Julian Moosmann, Marco Giordano, Christian Vogt, and Michele Magno. 2023. TinyissimoYOLO: A Quantized, Low-Memory Footprint, TinyML Object Detection Network for Low Power Microcontrollers. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, Hangzhou, China. https://doi.org/10.1109/aicas57966.2023.10168657

[56] Thorsten Ohl. 1995. Drawing Feynman diagrams with FX340-1 and METAFONT. *Computer Physics Communications* 90, 2–3 (Oct. 1995), 340–354. https://doi.org/10.1016/0010-4655(95)90137-s

[57] Jihyeok Park, Seungmin An, Dongjun Youn, Gyeongwon Kim, and Sukyoung Ryu. 2021. JEST: N+1-version Differential Testing of Both JavaScript Engines and Specification. arXiv:2102.07498 [cs.SE]

[58] Daniel Rosa, Filipe R. Cordeiro, Ruan Carvalho, Everton Souza, Sergio Chevtchenko, Luiz Rodrigues, Marcelo Marinho, Thales Vieira, and Valmir Macario. 2023. Recognizing Handwritten Mathematical Expressions of Vertical Addition and Subtraction. arXiv:2308.05820 [cs.CV]

[59] Francesco Rullani. 2006. *Dragging developers towards the core. How the Free/Libre/Open Source Software community enhances developers' contribution.* LEM Papers Series 2006/22. Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies, Pisa, Italy. https://ideas.repec.org/p/ssa/lemwps/2006-22.html

[60] Konstantin Serebryany, Derek Bruening, Alexander Potapenko, and Dmitriy Vyukov. 2012. AddressSanitizer: A Fast Address Sanity Checker. In *2012 USENIX Annual Technical Conference (USENIX ATC 12)*. USENIX Association, Boston, MA, 309–318. https://www.usenix.org/conference/atc12/technical-sessions/presentation/serebryany

[61] Sumeet S. Singh. 2018. Teaching Machines to Code: Neural Markup Generation with Visual Attention. arXiv:1802.05415 [cs.LG]

[62] Vidush Singhal, Akul Abhilash Pillai, Charitha Saumya, Milind Kulkarni, and Aravind Machiry. 2022. Cornucopia : A Framework for Feedback Guided Generation of Binaries. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22)*. ACM. https://doi.org/10.1145/3551349.3561152

[63] Pawan Sinha and Richard Russell. 2011. Perceptually-based Comparison of Image Similarity Metrics. *Perception* 40 (11 2011), 1269–81. https://doi.org/10.1068/p7063

[64] American Physical Society. 2018. *REVTeX 4.2 Author's Guide*. American Physical Society, Ridge, NY, USA.

[65] Thodoris Sotiropoulos, Stefanos Chaliasos, Vaggelis Atlidakis, Dimitris Mitropoulos, and Diomidis Spinellis. 2021. Data-Oriented Differential Testing of Object-Relational Mapping Systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 1535–1547. https://doi.org/10.1109/ICSE43902.2021.00137

[66] Evgeniy Stepanov and Konstantin Serebryany. 2015. MemorySanitizer: Fast detector of uninitialized memory use in C++. In *2015 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. 46–55. https://doi.org/10.1109/CGO.2015.7054186

[67] Chengnian Sun, Vu Le, and Zhendong Su. 2016. Finding and analyzing compiler warning defects. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 203–213. https://doi.org/10.1145/2884781.2884879

[68] Bezhentcev Roman Vadimovich. 2013. Software for creating pictures in the LaTeX environment. arXiv:1304.0600 [cs.GR]

[69] Domonkos F. Vamossy. 2023. Social Media Emotions and IPO Returns. arXiv:2306.12602 [q-fin.PR]

[70] Zelun Wang and Jyh-Charn Liu. 2019. Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training. arXiv:1908.11415 [cs.LG]

[71] Tsaihsuan Yang. 2017. An Intelligent and Automated PDF Format Checker. (2017).

[72] Xuejun Yang, Yang Chen, Eric Eide, and John Regehr. 2011. Finding and Understanding Bugs in C Compilers. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation* (San Jose, California, USA) *(PLDI '11)*. Association for Computing Machinery, New York, NY, USA, 283–294. https://doi.org/10.1145/1993498.1993532

[73] A. Zeller and R. Hildebrandt. 2002. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering* 28, 2 (2002), 183–200. https://doi.org/10.1109/32.988498

[74] Yingying Zheng, Wensheng Dou, Yicheng Wang, Zheng Qin, Lei Tang, Yu Gao, Dong Wang, Wei Wang, and Jun Wei. 2022. Finding bugs in Gremlin-based graph database systems via Randomized differential testing. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis* (<conf-loc>, <city>Virtual</city>, <country>South Korea</country>, </conf-loc>) *(ISSTA 2022)*. Association for Computing Machinery, New York, NY, USA, 302–313. https://doi.org/10.1145/3533767.3534409

[75] Fengmin Zhu and Fei He. 2022. EqFix: Fixing LaTeX Equation Errors by Examples. arXiv:2107.00613 [cs.PL]