The 26th Winona Computer Science Undergraduate Research Symposium

April 29, 2026

8:00 to 10:15am

https://minnstate.zoom.us/j/91438745171

Zoom Meeting ID: 914 3874 5171

Winona State University
Winona, MN

Sponsored by the Department of Computer Science

at Winona State University

WINONA STATE UNIVERSITY

Computer Science Department

http://cs.winona.edu

Table of Contents

Evaluating WAVE and Google Lighthouse on a Controlled WCAG 2.1 Violation Test Page

Saabiriin Abdi
Saabiriin46@gmail.com
Department of Computer Science
Winona State University
Winona, Minnesota, USA

Abstract

Automated accessibility evaluation tools, such as WAVE and Google Lighthouse, are widely used to assess compliance with the Web Content Accessibility Guidelines (WCAG). However, most existing evaluations rely on real-world websites, where the true number of accessibility violations is unknown, preventing calculation of accuracy metrics such as precision and recall. This study addresses these limitations by constructing a controlled HTML webpage that contains forty intentional WCAG 2.1 Level A and AA violations. WAVE and Lighthouse were executed forty times under identical conditions, and their outputs were compared against a predefined ground-truth data set. Results indicate that WAVE achieves higher detection coverage but produces a higher volume of non-actionable outputs, while Lighthouse identifies fewer violations yet demonstrates stronger performance on programmatically testable criteria, including Accessible Rich Internet Applications (ARIA) validation and contrast analysis. The findings confirm that no single automated tool provides comprehensive WCAG coverage and reinforces the necessity of combining automated evaluation with expert manual assessment. This study contributes a reproducible framework for quantitatively benchmarking accessibility tools under controlled conditions.

Keywords

Web accessibility, WCAG 2.1, automated evaluation, WAVE, Google Lighthouse, false negatives, rule coverage, Accessibility Testing

1 Introduction

Web accessibility ensures that individuals with disabilities can perceive, navigate, and interact with digital content. As education, healthcare, employment, and government services increasingly move online, inaccessible interfaces create significant barriers for users with visual, auditory, motor, and cognitive disabilities. The Web Content Accessibility Guidelines (WCAG) define the primary international standard for accessible web design, evolving from WCAG 2.0 (2008) to WCAG 2.1 (2018) and WCAG 2.2 (2023) to address mobile interaction, low vision accessibility, cognitive requirements, and modern web applications [1, 2].

The Web Content Accessibility Guidelines (WCAG) are organized around four foundational principles known as POUR: content must be Perceivable to users, Operable through multiple input methods, Understandable in both presentation and behavior, and Robust enough to be interpreted reliably by assistive technologies. These principles structure the guidelines and map to testable success criteria and conformance levels. Figure 1 illustrates this hierarchy. The controlled test page used in this study embeds forty intentional WCAG 2.1 Level A and AA violations distributed across all four principles, enabling systematic evaluation of detection coverage and tool sensitivity across accessibility categories.

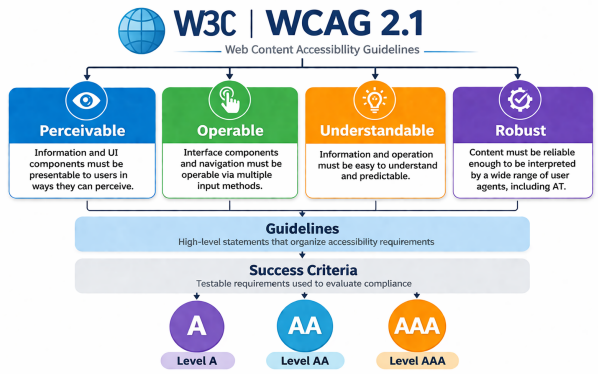

Figure 1: Overview of the WCAG 2.1 structure, showing the four POUR principles (Perceivable, Operable, Understandable, Robust), associated guidelines, and success criteria levels. This figure was created by the author.

Despite the maturity of WCAG, large-scale analyses such as the WebAIM Million consistently report that more than 96 percent of homepages contain detectable WCAG violations [3, 4]. Prior research has shown that automated accessibility tools differ substantially in rule coverage and detection behavior. Manca et al. identify transparency and reporting inconsistencies across evaluation engines [5], while Chadli et al. demonstrate that free and open-source tools vary significantly in accuracy and coverage [6]. Ara and Sik-Lanyi further report substantial divergence in detection behavior across automated tools [7]. Collectively, these findings highlight the need for multi-tool evaluation and expert manual inspection.

A central limitation in existing research is that most evaluations rely on real-world websites where the true number of accessibility violations is unknown. Without ground truth, it is not possible to compute false negatives or assess detection reliability in a controlled manner, which are essential metrics for evaluating tool performance [8]. Early foundational work first identified persistent disagreement among automated tools [9, 10], and more recent studies confirm that this limitation persists in modern evaluation engines [5]–[7].

Proceedings of the 26th Winona Computer Science Undergraduate Research Seminar, Winona, MN, US
2026.

This study addresses this gap by constructing a controlled HTML webpage containing forty intentional WCAG 2.1 Level A and AA violations. By embedding violations across all four POUR principles, the study creates a reproducible dataset that enables systematic measurement of detection coverage, determinism, and cross-tool agreement. Based on prior research, it is hypothesized that both WAVE and Google Lighthouse will demonstrate consistent behavior across repeated executions but will differ in detection coverage. WAVE is expected to identify a broader range of violations due to its heuristic-based scanning approach [11, 12], whereas Lighthouse is expected to perform more strongly on strictly machine-verifiable criteria such as ARIA validation and contrast analysis [13].

The primary contribution of this research is a controlled and reproducible evaluation framework that enables automated accessibility tools to be benchmarked against a known violation set. This framework supports computation of detection coverage and false-negative counts, which is not feasible when evaluating real-world websites without ground truth. The study advances understanding of automated accessibility evaluation and reinforces the need for hybrid testing approaches that combine automated tools with expert manual assessment [5, 7, 13].

2 Background

2.1 Automated Accessibility Evaluation Tools

Automated accessibility evaluation tools play a central role in large-scale assessments of WCAG conformance. Tools such as WAVE and Google Lighthouse are widely used in research and practice to identify issues detectable through rule-based analysis. WAVE has been examined extensively in both historical and contemporary accessibility studies [10], [18], while Lighthouse is frequently included in multi-tool evaluations due to its integration with Chrome DevTools and its emphasis on programmatically testable criteria [13–15].

Despite their widespread use, these tools can assess only a subset of WCAG requirements. Many criteria depend on semantic interpretation, interaction context, or user experience, which cannot be evaluated through automated means. Prior work has shown that such tools differ in their rule sets, detection logic, and interpretation of WCAG success criteria, resulting in inconsistent reporting across evaluation tools [5, 7]. These constraints underscore the need for careful methodological design when using automated tools in empirical research.

2.2 Limitations of Existing Evaluation

A substantial body of research has documented the variability and inconsistency of automated accessibility tools. Benchmarking studies show that different tools often disagree on both the number and type of issues detected, even when analyzing the same webpage [5, 6]. Comparative evaluations further reveal gaps in rule coverage, differences in how WCAG criteria are operationalized, and sensitivity to HTML structure and ARIA usage [7, 8]. These challenges complicate efforts to compare tools directly or rely on automated results as definitive indicators of accessibility quality.

A critical methodological constraint is the absence of ground truth datasets in most evaluations. Studies that analyze real world websites cannot determine the true number of accessibility violations, which prevents calculation of accurate metrics such as precision, recall, and false positive rates. Several authors have emphasized the need for controlled testing environments where violations are explicitly defined and measurable [8, 13, 16]. Machine learning approaches have attempted to predict accessibility barriers or estimate coverage error, but these models depend on labeled datasets and often inherit inconsistencies from the tools used to generate training data [14, 17, 18]. As a result, controlled experimental designs remain essential for evaluating the capabilities and limitations of automated tools.

2.3 Need for Controlled Testing

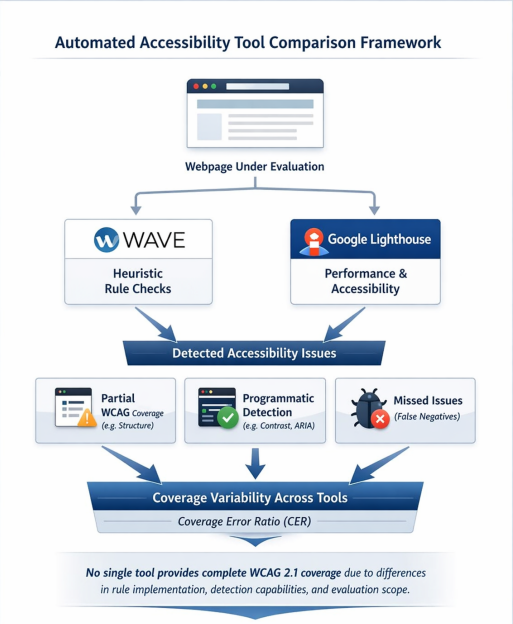

Figure 2: Comparison framework for WAVE and Google Lighthouse.

Controlled testing environments address the methodological gaps associated with real world evaluations by enabling measurement against a known set of accessibility violations. Constructing a webpage with a predefined ground truth dataset enables computation of detection coverage, precision, recall, and false positive rates, which are not feasible when the true number of violations is unknown.

Large scale studies such as the WebAIM Million highlight the widespread presence of accessibility barriers across the web [3, 3], but they also demonstrate the limits of relying solely on automated tools for comprehensive evaluation. Similar findings in institutional website analyses reinforce the need for controlled experiments that isolate tool behavior from the variability of real world content [19].

Figure 2 illustrates this challenge by showing how different tools diverge in rule coverage, detection capabilities, and evaluation scope. These discrepancies underscore the importance of controlled methodologies that enable systematic examination of tool performance, including determinism and cross-tool agreement.

Multi-tool evaluations have shown that automated tools may produce inconsistent results depending on execution context, browser configuration, or rendering conditions [12, 15]. Controlled testing provides the necessary stability to measure these behaviors rigorously.

3 Methodology

3.1 Study Design

This study employed a controlled experimental design to evaluate the detection capabilities of WAVE and Google Lighthouse with respect to WCAG 2.1 Level A and AA success criteria. A controlled environment was necessary because real-world websites do not provide a verifiable ground truth, making it impossible to compute key accuracy metrics such as precision, recall, and false-negative rates. Prior research consistently highlights this limitation and emphasizes the need for reproducible test environments when benchmarking automated accessibility tools [4–6, 8].

To address this gap, a custom HTML webpage was constructed containing forty intentional WCAG 2.1 violations. The number forty was selected to balance methodological rigor with analytical manageability while ensuring representation across a diverse set of accessibility categories. Violations were distributed across the four POUR principles: Perceivable, Operable, Understandable, and Robust—to capture the breadth of WCAG requirements and to enable principle-level analysis of tool performance. Figure 3 summarizes the structured distribution of these intentional violations within the controlled test page.

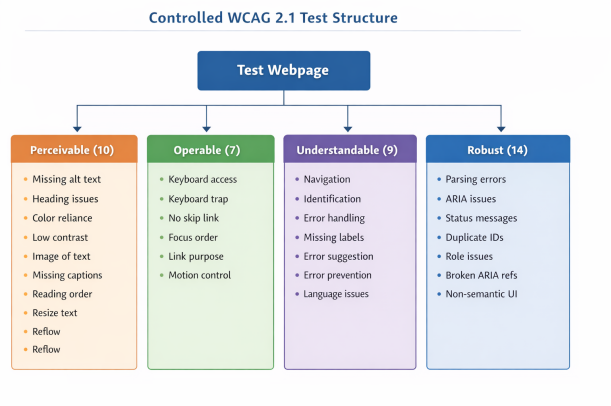

Figure 3: Distribution of intentional WCAG 2.1 violations across the POUR principles.

3.2 Ground Truth Dataset Construction

The controlled HTML test page was designed to embed a wide range of WCAG 2.1 violations representative of common accessibility failures observed on real world websites [3, 3]. Violations included missing alternative text, insufficient color contrast, incorrect ARIA roles, unlabeled form fields, keyboard inaccessibility, and structural markup errors as seen in Figure 3.

Each violation was mapped to its corresponding WCAG 2.1 success criterion to ensure traceability. Violations were also classified by detectability:

- **Automatable**: Criteria detectable through rule-based or programmatic checks.
- **Partially Automatable**: Criteria requiring both automated detection and human interpretation.
- **Non-automatable**: Criteria dependent on human judgment, such as meaningful sequence or descriptive link text.

This classification aligns with prior work distinguishing machine testable and human dependent criteria [13, 16]. The complete list of intentional WCAG 2.1 violations, including their identifiers, success criteria mappings, and descriptions, is provided in Appendix A.

3.3 Experimental Environment

All evaluations were conducted in a standardized environment to ensure reproducibility and minimize confounding variables. The configuration included:

- **Operating system:** Windows 11 (23H2)
- **Browser:** Opera One (Chromium-based), version 108
- **Screen resolution:** 1920×1080
- **Google Lighthouse version:** 11.2.0 (CLI mode)
- **WAVE version:** WAVE API 3.1 (2024)

The browser operated in incognito mode with caching disabled, hardware acceleration enabled, and no extensions or background applications. A controlled environment is essential because automated accessibility tools can produce inconsistent results depending on browser configuration, rendering behavior, and execution context [12, 15].

3.4 Execution Procedure

WAVE and Lighthouse were each executed forty times under identical conditions to measure detection coverage and determinism. All executions were performed manually following a strictly controlled and repeatable procedure to ensure consistency across runs. A strict no-interaction policy was enforced during every audit: no scrolling, clicking, or dynamic content interaction occurred, ensuring that both tools evaluated the page in a consistent static state.

During each execution, WAVE generated a report containing errors, alerts, structural issues, and ARIA-related findings generated a report containing accessibility issues consistent with patterns observed in automated accessibility evaluation tools [20], while Lighthouse produced an accessibility score and detailed audit results for contrast, ARIA usage, keyboard accessibility, and related criteria [16]. To situate this procedure within the broader study design, Figure 4 illustrates the complete experimental workflow, from construction of the controlled test page and ground-truth mapping to repeated tool execution, detection logging, and quantitative analysis.

3.5 Ground Truth Mapping and Evaluation Metrics

Each issue reported by WAVE and Lighthouse was systematically mapped to the predefined ground-truth dataset to assess detection performance. This mapping enabled classification of tool outputs using three standard categories:

- **True Positive (TP):** A reported issue that correctly corresponds to a predefined violation.

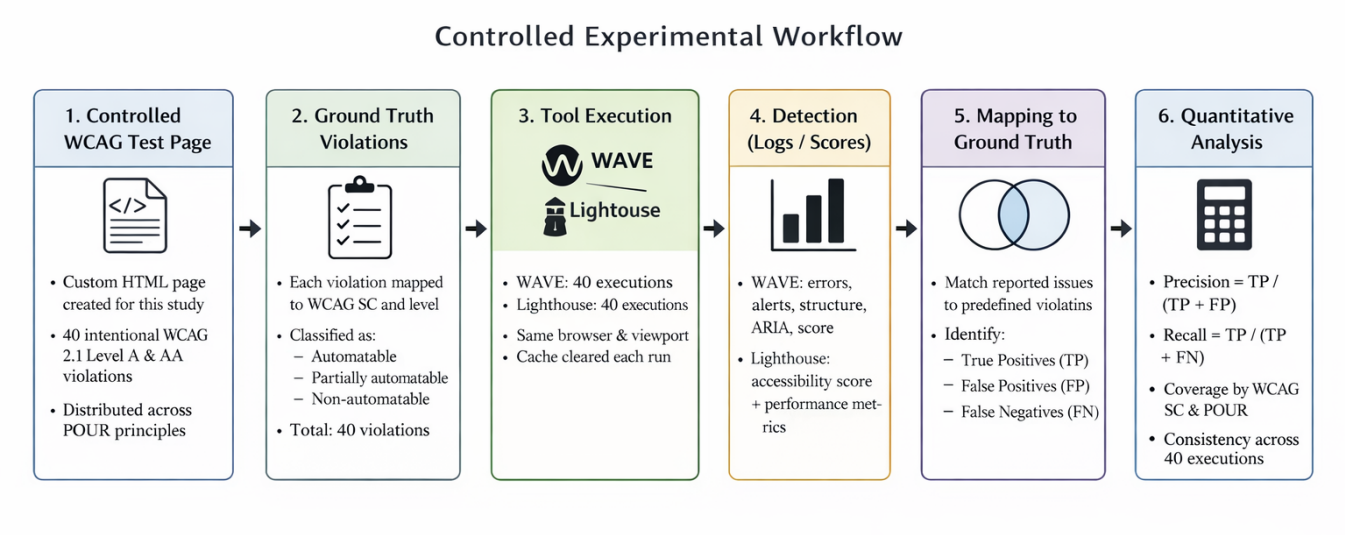

Figure 4: Overview of the controlled experimental workflow used in the study.

- **False Negative (FN):** A predefined violation that was not detected by the tool.
- **False Positive (FP):** A reported issue that does not correspond to any predefined violation.

The controlled test page was intentionally designed to minimize false positives arising from page structure or ambiguous markup. However, some tool-specific discrepancies may still occur due to differences in rule implementation, interpretation of WCAG criteria, and ARIA processing behavior [5, 7]. All mappings were manually validated to ensure consistency and accuracy. Based on these classifications, the following evaluation metrics were defined:

- **Detection Coverage:** The proportion of the forty predefined violations detected by each tool.
- **Precision:**

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:**

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Determinism (Stability Coefficient):** The proportion of identical outputs across forty executions, where a value of 1.0 indicates perfect consistency.
- **Cross-Tool Overlap:** The number of violations detected by both tools, providing insight into shared detection capability and complementary strengths.

This evaluation framework supports systematic comparison of automated accessibility tools and aligns with established multi-tool benchmarking approaches in accessibility research [12, 20].

3.6 Validity Considerations

Although the study employs a controlled and reproducible design, several validity considerations are acknowledged to contextualize the findings.

Internal Validity. Internal validity is supported through a fixed execution environment, consistent procedures, and a predefined ground-truth dataset. These controls reduce confounding variables, ensuring that observed differences primarily reflect tool behavior. However, manual execution introduces a minor risk of human error despite strict adherence to a no-interaction protocol.

External Validity. The results may not fully generalize to real-world websites, which often include dynamic content, interactive elements, and complex structures [19]. While controlled environments enable precise measurement, they do not capture the full variability of real-world accessibility barriers.

Construct Validity. Construct validity is limited by the use of a static test page containing forty intentional WCAG 2.1 violations. Although these span all POUR principles, they cannot represent the full diversity of accessibility issues, particularly those requiring user interaction or semantic interpretation.

Conclusion Validity. The evaluation metrics (recall, detection coverage, determinism, and overlap) provide a structured comparison of tool performance. However, they depend on accurate ground-truth mapping, and some classification ambiguity may remain for partially automatable criteria.

Tool and Browser Bias. Differences in rule coverage, WCAG interpretation, and detection logic between WAVE and Lighthouse may introduce systematic bias [5–7]. In addition, browser rendering differences can affect accessibility tree generation [15]. Although a standardized browser configuration was used, minor variability may still occur.

4 Results

Empirical findings from forty executions of WAVE and Google Lighthouse on a controlled WCAG 2.1 violation test page are reported. The analysis examines detection coverage, recall, false-negative rate, determinism, cross-tool overlap, and performance across the four

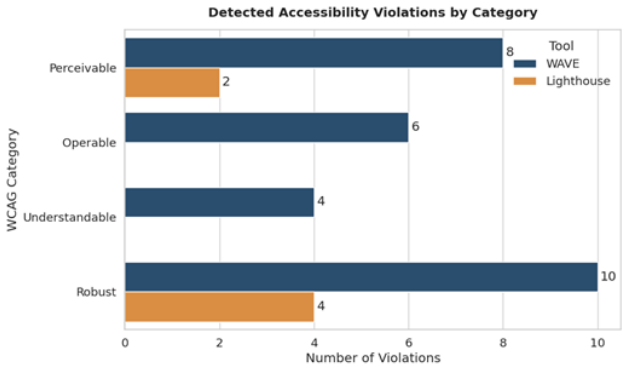

Figure 5: Detected WCAG 2.1 violations by POUR category for WAVE and Lighthouse.

POUR principles, based on comparison with a predefined ground-truth dataset of forty intentional accessibility violations.

4.1 Detection Coverage

Detection coverage measures the proportion of the forty intentional WCAG 2.1 violations correctly identified by each tool. Table 1 summarizes the total number of violations, detected violations, missed violations, and overall coverage for both WAVE and Google Lighthouse.

Table 1: Detection coverage for WAVE and Lighthouse

Tool	Total Violations	Detected	Missed	Coverage (%)
WAVE	40	22	18	55%
Lighthouse	40	12	28	30%

WAVE achieved a higher detection coverage (55%) than Google Lighthouse (30%), identifying a substantially larger proportion of the intentionally injected accessibility violations. This suggests that WAVE provides broader issue detection across the test dataset, whereas Lighthouse demonstrates a more limited detection scope.

Figure 5 further disaggregates detected violations across the WCAG 2.1 POUR categories. WAVE identified issues across all four categories, with the highest counts in the Robust (10) and Perceivable (8) dimensions, followed by Operable (6) and Understandable (4). In contrast, Lighthouse detections were concentrated primarily in the Robust category (4), with minimal detection in Perceivable (2) and no detected issues in either the Operable or Understandable categories.

Overall, WAVE exhibited more comprehensive coverage across the POUR principles, while Lighthouse showed a narrower detection profile focused primarily on structural and machine-readable accessibility checks. This pattern reflects differences in tool design: WAVE's heuristic-based scanning approach captures a broader range of accessibility failures, whereas Lighthouse's rule-based engine prioritizes a more constrained subset of programmatically testable criteria.

4.2 Recall and False Negative Rate

Detection coverage was computed using the ground-truth dataset and measured the proportion of true WCAG 2.1 violations successfully detected by each tool. The false-negative rate was defined as the proportion of violations present in the dataset but not identified by the tools.

WAVE achieved higher recall (55%) compared to Google Lighthouse (30%), indicating greater sensitivity to the predefined violation set. Correspondingly, WAVE exhibited a lower false-negative rate (45%), while Lighthouse showed a higher false-negative rate (70%), reflecting reduced overall detection coverage across the test dataset.

Precision was defined within the evaluation framework; however, it was not computed in this study. This is because the experimental design did not include a systematic classification of all tool-generated outputs into true-positive and false-positive categories. As a result, false-positive counts could not be reliably established, preventing the calculation of precision.

Overall, the results indicate a trade-off between detection coverage and strictness of evaluation. WAVE demonstrates broader detection across WCAG 2.1 categories, capturing a larger proportion of injected violations, whereas Lighthouse produces more constrained outputs focused primarily on programmatically verifiable accessibility criteria.

Table 2: Recall and False-Negative Rates for Automated Tools

Tool	Recall	False Negative Rate
WAVE	55%	45%
Lighthouse	30%	70%

4.3 Cross-Tool Overlap

Cross-tool overlap was quantified as the proportion of violations detected by both tools relative to the total number of unique detected violations. Out of the combined detection set, 9 violations were identified by both WAVE and Lighthouse, corresponding to a cross-tool overlap of approximately 29%. This overlap was primarily concentrated in clearly defined accessibility issues such as missing form labels and color contrast violations, indicating partial agreement on strictly machine-verifiable criteria.

WAVE uniquely detected a larger number of violations, particularly those related to structural and contextual accessibility issues that require broader heuristic interpretation. In contrast, Lighthouse uniquely identified a smaller set of issues, primarily associated with ARIA validation and rule-based checks.

These differences reflect variations in rule implementation, evaluation scope, and interpretation of WCAG success criteria across tools [5, 7]. The limited overlap suggests that the tools provide complementary rather than redundant coverage, with each tool capturing distinct subsets of accessibility issues.

4.4 Determinism

Determinism was evaluated using a stability coefficient defined as the proportion of identical outputs across forty executions per tool. Both tools demonstrated high determinism under controlled conditions. Lighthouse produced identical outputs across all runs, indicating fully stable behavior. WAVE exhibited minor variability in alert-level classification across executions; however, the set of detected violations remained consistent. Overall, these results indicate that both tools are stable under repeated evaluations, with only minimal variation observed in non-critical classification layers.

Figure 6 presents a word cloud summarizing WCAG violation categories detected by WAVE and Lighthouse across all runs. Term size reflects relative frequency within the combined outputs of both tools. Prominent terms such as "ARIA," "Missing," and "Contrast" indicate frequently detected issue types, consistent with a focus on programmatically verifiable criteria, while smaller terms represent less frequently identified or tool-specific violations. Although qualitative, the visualization complements the quantitative findings by highlighting overall detection patterns.

Figure 6: Word cloud representation of WCAG violation types detected by WAVE and Lighthouse across forty runs.

4.5 Detection Across POUR Categories

Violations were grouped according to the WCAG POUR principles. WAVE identified a greater number of issues across the Perceivable, Operable, and Understandable categories, reflecting its heuristic-based detection approach. In contrast, Lighthouse exhibited comparatively stronger performance in the Robust category, particularly in ARIA validation and programmatic correctness checks. Despite these differences, both tools demonstrated reduced detection capability in categories requiring contextual interpretation, which remain challenging for automated accessibility evaluation systems.

5 Conclusion

This study evaluated the effectiveness of two widely used automated accessibility evaluation tools, WAVE and Google Lighthouse, under a controlled WCAG 2.1 test environment containing forty intentionally embedded Level A and AA violations. By establishing a predefined ground-truth dataset, the study enabled systematic measurement of detection coverage (recall-like performance), cross-tool agreement, and output consistency—metrics that are difficult to obtain in real-world website evaluations due to the absence of known violation baselines.

The results demonstrate a clear difference in detection behavior between the two tools. WAVE achieved higher detection coverage (55%) compared to Lighthouse (30%), indicating broader detection capability across multiple WCAG dimensions, particularly structural and content-level issues. In contrast, Lighthouse exhibited more conservative detection behavior, focusing primarily on strictly machine-verifiable conditions such as ARIA validation and contrast-related checks. This difference reflects fundamental variations in evaluation design, where heuristic-based scanning (WAVE) produces broader coverage, while rule-based auditing (Lighthouse) prioritizes consistency in machine-verifiable conditions.

Across all analyses, both tools demonstrated consistent internal behavior under repeated executions, with Lighthouse showing fully deterministic outputs and WAVE exhibiting only minor variation in classification labels. However, cross-tool overlap remained limited, with agreement primarily occurring on straightforward violations such as missing labels and contrast failures. This suggests that each tool captures different subsets of WCAG violations, reinforcing their complementary rather than substitutive roles in accessibility assessment workflows.

A key methodological contribution of this study is the use of a controlled violation dataset that enables computation of detection coverage and false-negative counts. Unlike prior studies that rely on real-world websites with unknown ground truth, this approach provides a reproducible framework for benchmarking accessibility tools under identical evaluation conditions. At the same time, the study is limited by the absence of full false-positive annotation, which prevented computation of precision and related composite metrics such as F1-score.

Overall, the findings highlight that no single automated tool provides comprehensive WCAG 2.1 coverage. WAVE offers broader detection coverage, while Lighthouse provides more constrained but stable rule-based evaluation. These complementary strengths support the continued use of multi-tool evaluation strategies combined with expert manual review to achieve more complete accessibility assessment. Future work should extend this framework by incorporating full confusion-matrix labeling, expanding the violation set to include WCAG 2.2 criteria, and evaluating additional automated tools to further generalize findings across accessibility evaluation ecosystems.

References

[1] W3C. Web content accessibility guidelines (wcag) 2.1. https://www.w3.org/TR/WCAG21/, 2018. Accessed: 2026-04-26.

[2] W3C. Web content accessibility guidelines (wcag) 2.2. https://www.w3.org/TR/WCAG22/, 2023. Accessed: 2026-04-26.

[3] WebAIM. The webaim million: 2024 report. https://webaim.org/projects/million/2024, 2024. Accessed: 2026-04-26.

[4] WebAIM. The webaim million: 2023 report. https://webaim.org/projects/million/2023, 2023. Accessed: 2026-04-26.

[5] Marco Manca, Vanessa Palumbo, Fabio Paternò, and Carmen Santoro. The transparency of automatic web accessibility evaluation tools: Design criteria, state of the art, and user perception. *ACM Transactions on Accessible Computing*, 16(1), 2023.

[6] Fatima Ezzahra Chadli, Driss Gretete, and Aniss Moumen. Comparison of free and open source wcag accessibility evaluation tools. In *Proceedings of NISS '23*, 2023.

[7] J. Ara and C. Sik-Lanyi. Automated evaluation of accessibility issues of webpage content: Tool and evaluation. *Scientific Reports*, 15:9516, 2025.

[8] Mukesh Rajmohan, Smit Desai, and Sanchari Das. Multi-tool analysis of user interface & accessibility in deployed web-based chatbots. In *Proceedings of CUI '25*, 2025.
[9] Giorgio Brajnik. A comparative test of web accessibility evaluation methods. In *Proceedings of Assets '08*, 2008.
[10] Rita Ismailova and Yavuz Inal. Comparison of online accessibility evaluation tools: An analysis of tool effectiveness. *IEEE Access*, 10:58233–58239, 2022.
[11] A. Alsaeedi. Comparing web accessibility evaluation tools and evaluating the accessibility of webpages: Proposed frameworks. *Information*, 11(1):40, 2020.
[12] Pooja Naresh Bhatia and Sam Malek. A historical review of web accessibility using wave. In *Proceedings of GE@ICSE '24*, 2024.
[13] Carlos Alberto Silva, Arthur F. B. A. de Oliveira, Delvani Antônio Mateus, Heitor Augustus Xavier Costa, and André Pimenta Freire. Types of problems encountered by automated tool accessibility assessments, expert inspections and user testing: a systematic literature mapping. In *Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*, IHC '19, pages 1–11, New York, NY, USA, 2019. Association for Computing Machinery.
[14] Shadi Abou-Zahra, Judy Brewer, and Michael Cooper. Ai for web accessibility: Is conformance evaluation a way forward? In *Proceedings of W4A '18*, 2018.
[15] Stacy M. Branham et al. The state of digital accessibility. *Communications of the ACM*, 68(4):38–41, 2025.
[16] Carlos Alberto Silva et al. Systematic literature mapping of accessibility evaluation methods. In *Proceedings of IHC '19*, 2019.
[17] Markel Vigo, Justin Brown, and Vivienne Conway. Benchmarking web accessibility evaluation tools. In *Proceedings of W4A '13*, 2013.
[18] Google. Lighthouse accessibility audit documentation, 2024. Accessed: 2026-04-26.
[19] Syed Fatiul Huq, Ziyao He, Yirui He, and Sam Malek. Bridging the gap between automated intervention and actual user experience: A mixed-methods study on mobile accessibility issues for screen reader users. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, CHI '26, pages 1–27, New York, NY, USA, 2026. Association for Computing Machinery.
[20] Katelyn Leedy et al. Accessibility in the age of generative ai web builders. In *Proceedings of Mindtrek '25*, 2025.

A WCAG 2.1 Violations in Controlled Test Page

This appendix lists the forty intentional WCAG 2.1 violations embedded in the controlled test page. Violations are organized by POUR principles and AAA-level criteria for clarity and interpretability.

A.1 Perceivable Violations

- **P-1.1.1-1 (1.1.1 Non-text Content, Level A)**: Image missing alternative text.
- **P-1.2.2-1 (1.2.2 Captions, Level A)**: Video without captions.
- **P-1.3.1-1 (1.3.1 Info and Relationships, Level A)**: Incorrect heading structure (skipped levels).
- **P-1.3.2-1 (1.3.2 Meaningful Sequence, Level A)**: Content presented in incorrect logical order.
- **P-1.4.1-1 (1.4.1 Use of Color, Level A)**: Information conveyed using color only.
- **P-1.4.3-1 (1.4.3 Contrast (Minimum), Level AA)**: Insufficient text contrast.
- **P-1.4.4-1 (1.4.4 Resize Text, Level AA)**: Text does not scale appropriately.
- **P-1.4.5-1 (1.4.5 Images of Text, Level AA)**: Image used instead of accessible text.
- **P-1.4.10-1 (1.4.10 Reflow, Level AA)**: Fixed layout prevents responsive reflow.
- **P-1.4.11-1 (1.4.11 Non-text Contrast, Level AA)**: UI component lacks sufficient contrast.

A.2 Operable Violations

- **O-2.1.1-1 (2.1.1 Keyboard, Level A)**: Interactive element not keyboard accessible.
- **O-2.1.2-1 (2.1.2 No Keyboard Trap, Level A)**: Keyboard navigation is blocked.
- **O-2.2.2-1 (2.2.2 Pause, Stop, Hide, Level A)**: Moving content cannot be paused.
- **O-2.4.3-1 (2.4.3 Focus Order, Level A)**: Illogical tab order.
- **O-2.4.4-1 (2.4.4 Link Purpose, Level A)**: Link text is not descriptive ("Click here").
- **O-2.5.3-1 (2.5.3 Label in Name, Level A)**: Visible label does not match accessible name.

A.3 Understandable Violations

- **U-3.1.2-1 (3.1.2 Language of Parts, Level A)**: Language not specified for content.
- **U-3.2.1-1 (3.2.1 On Focus, Level A)**: Unexpected context change on focus.
- **U-3.2.2-1 (3.2.2 On Input, Level A)**: Unexpected navigation on input.
- **U-3.2.3-1 (3.2.3 Consistent Navigation, Level AA)**: Navigation structure is inconsistent.
- **U-3.2.4-1 (3.2.4 Consistent Identification, Level AA)**: Components labeled inconsistently.
- **U-3.3.1-1 (3.3.1 Error Identification, Level A)**: Errors not identified to user.
- **U-3.3.2-1 (3.3.2 Labels or Instructions, Level A)**: Form input lacks label.
- **U-3.3.3-1 (3.3.3 Error Suggestion, Level AA)**: No suggestions for input errors.
- **U-3.3.4-1 (3.3.4 Error Prevention, Level AA)**: No safeguards for form submission.

A.4 Robust Violations

- **R-4.1.1-1 (4.1.1 Parsing, Level A)**: Improper HTML nesting.
- **R-4.1.2-1 (4.1.2 Name, Role, Value, Level A)**: Invalid ARIA attribute value.
- **R-4.1.2-2 (4.1.2 Name, Role, Value, Level A)**: Non-semantic interactive element.
- **R-4.1.2-3 (4.1.2 Name, Role, Value, Level A)**: Missing ARIA role.
- **R-4.1.2-4 (4.1.2 Name, Role, Value, Level A)**: Broken ARIA reference.
- **R-4.1.3-1 (4.1.3 Status Messages, Level AA)**: Dynamic updates not announced.

A.5 AAA-Level Violations

- **AAA-1.2.7-1 (1.2.7 Extended Audio Description)**: Video lacks extended audio description.
- **AAA-3.1.5-1 (3.1.5 Reading Level)**: Content exceeds accessible reading level.
- **AAA-3.1.6-1 (3.1.6 Pronunciation)**: Ambiguous pronunciation not clarified.

The Linguistic Barrier of Data Disclosures

Garrett Buryska
Winona State University
Winona, Minnesota
gburyska@protonmail.com

Abstract

Websites and online services have become essential to daily life, and with that, guardrails in the form of privacy policies are as important as ever to maintain user consent. However, privacy policies are one-sided. While these policies are meticulously crafted by corporations, the resulting documents are often dense and legally complex for the average user. This study utilized natural language processing to investigate the relationship between readability and the presence of terms related to data handling. The dataset used for the analysis included 135 privacy policies across many sectors. A disclosure frequency score was developed using the Spacy Python library to detect disclosures of data sharing, tracking, and sensitive information using common keywords while also accounting for negation. Readability was scored using the Textstat library, primarily utilizing the Flesch-Kincaid Grade Level. Statistical analysis revealed a weak positive correlation between grade level and the disclosure frequency score, showing that policies with possibly more invasive data practices are more difficult to read. A Wilcoxon Signed-Rank Test also showed that sentences containing data disclosure keywords are significantly harder to read than the rest of the text. Our results showed that text is less readable around data handling disclosures, possibly concealing important privacy disclosures from the average user.

CCS Concepts

• **Security and privacy** → **Usability in security and privacy**; • **Human-centered computing** → *Empirical studies in HCI*; • **Applied computing** → Law.

Keywords

Privacy Policies, Readability, Privacy Paradox, Text Analysis, Disclosure Frequency

ACM Reference Format:
Garrett Buryska. 2026. The Linguistic Barrier of Data Disclosures. In *Proceedings of the 26th Winona Computer Science Undergraduate Research Seminar*. ACM, New York, NY, USA, 5 pages.

1 Introduction

Today's internet functions as the modern town square, exchanging user data for services and relying on privacy policies as the legal means for obtaining consent between corporations and users. The current world depends on websites and applications for communication, commerce, and entertainment, making clear, accessible data-handling agreements essential. Even though they are important, research confirms that privacy policies remain notoriously difficult to read for the average adult, let alone children.

the 26th Winona Computer Science Undergraduate Research Seminar, Winona, MN, US 2026.

Privacy agreements prior to the Internet usually involved two parties establishing boundaries between private life and public activity, with both comprehending the terms and participating in decisions about them. Now, modern interactions between corporations and users are one-sided, where consumers can either accept the terms or lose the service [14]. Because of this, policy lengths are often lengthy to guarantee compliance with laws from the many different jurisdictions. This, in turn, results in a "privacy paradox": a user who is interested in their privacy fails to read the policy [7].

While length is a big, if not the main, factor producing the privacy paradox, the piece that tags right along with length is the hard-to-read nature of the policies. As privacy policies became longer, they also became more complex [7]. On average, a reader must have graduated from high school or completed some college, requiring approximately 13 to 16 years of formal education, to understand these documents [1]. Study [7] found that although only 18.4% of participants reported difficulty understanding policies, 55% did not actually understand them, showing a gap between perceived and actual comprehension.

Total text length does not increase most of the readability metrics' scores, so why is there an increase in complexity as length increases? This study set out to answer that question. As mentioned before, policies are often quite comprehensive, especially after the passage of the European Union's General Data Protection Regulation, which increased policy length by 325% [3]. With so many required disclosures, this study focused on whether readability is associated with disclosure frequency. While studies have examined readability across several sections of policies and by length, none have specifically analyzed readability based on disclosure occurrence, both intra- and inter-policy. Therefore, by investigating the relationship between disclosures and policy readability, this study sought to fill this gap and identify new possible causes of complexity in privacy policies.

In this study, we hypothesize that sentences containing disclosure keywords have a higher Flesch-Kincaid Grade Level than sentences without such keywords. We secondly hypothesize that there is a statistically significant positive relationship between the frequency of disclosure keywords and the Flesch-Kincaid Grade Level of corporate privacy policies.

Collecting the policies immediately posed a problem due to the lack of standardization of privacy policies worldwide. Most companies publish their privacy policies as HTML pages with no way to save them as normal text documents, so the study required a cleaning step to allow examination of the policies. To investigate the relationship, three variables needed to be collected for each sampled policy: overall policy readability score, the readability difference between sentences with and without disclosure keywords, and a custom score based on disclosure keyword frequency. To collect these variables, Python and many of its libraries were used.

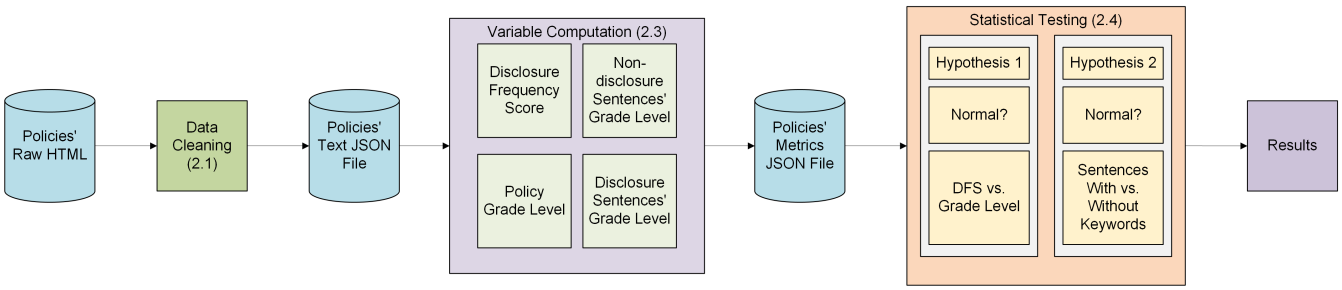

Figure 1: Methodology Diagram

After collecting these variables, a statistical analysis was conducted to test the study's two hypotheses. The results of this analysis provide further insight into how disclosure practices may affect privacy notice complexity, identifying a potential area for regulatory change.

2 Methodology

Since 2000, the average policy length has quadrupled [15]. Length increases coincide with new privacy policy regulations. In 2018, the General Data Protection Regulation coincided with a 928-word increase in the average length of policies that mention it [15]. In 2021, the average Flesch Reading Ease of privacy policies in a corpus of more than 50000 privacy policies was around 31, comparable to that of the Harvard Law Review. Interestingly, that score is 6 points below 2001 and 3 points below 2011 [15]. According to [15] and [4], scores above 60 are considered plain English, while scores below 30 are best understood by college graduates. Other studies have confirmed that readability is well below average [4–6, 11, 15].

With the regulations came new disclosure requirements, which explain the increase in length but not the decrease in readability unless the increase is related to the disclosures. Study [1] shows that different disclosure categories are very difficult to read, but it does not compare them with the rest of the policy sample, making it impossible to examine the relationship between disclosures and readability. The language used in legal disclosures to describe technical activities is often complex [10], which makes the hypotheses of this study intriguing. If the disclosures themselves are complex, the text within and directly around the disclosure could partially account for the policies' advanced readability.

The methodology followed in this study is outlined in Figure 1. Figure 1 visually maps to the following subsections to guide the reader: the initial Data Cleaning process is detailed in Section 2.1, the Variable Computation for our metrics is covered in Section 2.3, and the final Statistical Testing phase is explained in Section 2.4.

2.1 Dataset

To scientifically confirm the two hypotheses, a large dataset comprising 153 policies was collected. The dataset spanned a wide variety of industries. Due to the lack of standardization [2] and companies' reluctance to adopt standards [13], there is no template or standard format for privacy policies. There is also no requirement that privacy policies be in a document form. Because of those issues, the cleaning of the dataset was challenging. The selected privacy policies' HTML pages were saved. Tags that are not usually part of the body of privacy policies were automatically removed, as were headers. From the remaining HTML, only text from components commonly containing body text was used. Using the NLTK Python library, the text was tokenized into sentences, preserving proper sentence boundaries (including abbreviation recognition). Policies exceeding 200 words were combined and saved to a JSON file.

2.2 Readability Metric

There are many popular readability metrics, and while each could be used for this study, [5] found that they all correlate with one another. For ease, a singular readability metric was selected: the Flesch-Kincaid Grade Level [9]. Equation (1) is how the Flesch-Kincaid Grade Level is calculated. The score was generated using the Textstat Python library, which takes the given text as input and returns the grade level. The metric is easy to understand because the result indicates the grade level required to understand it. So, a score of 9-12 would be high school, 13-17 would be college, and above 17 would be postgraduate. The typical American reads at a 7th- to 8th-grade level [16].

$$GL = .39(AverageSentenceLength) + 11.8(Syllables/Word) - 15.59 \quad (1)$$

2.3 Algorithm to Compute Variables

To calculate the disclosure frequency score, a Python algorithm was created utilizing the Spacy natural language processing library. The Python library utilized sentence boundary detection, tokenization, and lemmatization. A set of about 15 keywords commonly included in disclosures was used. The algorithm did more than just count keyword occurrences. Using the spaCy library, it iterated through the sentences. Each sentence was checked for keywords from the categories of keywords found in Table 1. If found, a value was added to the keyword score depending on the keyword category. The sentence was also appended to the sentences used to calculate the difference in readability between sentences with and without keywords. Some jurisdictions require companies to declare what they do not do with the data, so negation was checked using Spacy and common negation strings. Negation, when present, resulted in the value being subtracted rather than added. If keywords from more than one category were used in a sentence, the keyword score was multiplied slightly. The score was then divided by the word

Table 1: Disclosure Keywords

Type	Keywords
Sensitive Data	Biometric, Health, Medical, Social Security, Precise Location
Partners	Sell, Monetize, Commercialize, Share, Transfer, Disclose
	Advertising, Marketing, Analytics, Brokers, Third Parties

Table 2: Shapiro-Wilk Normality Test

Variable	P-Value
Reading Grade Level	*0.1603
Disclosure Score	*0.5937
Sentence Differences	0.0003

* Indicates a normal distribution

count and multiplied by 100 to give the final disclosure frequency score, as shown in Equation (2). The grade level for the whole policy and for the grouped sentences is computed at the end of the algorithm, and all variables are saved to a JSON file.

$$DFS = (keywordscore/wordcount) \times 100 \quad (2)$$

2.4 Statistical Analysis

With the variables now housed in an easy-to-access, easily manipulated format, the next step is to run the actual statistical tests. Both hypotheses look at the relationship between readability and disclosure keywords. For each statistical analysis, the variables were checked for normality using the Shapiro-Wilk test [8, 12] before selecting a parametric or nonparametric test. The first hypothesis investigated whether policies with higher disclosure scores corresponded to a higher required grade level for understanding. Depending on the results of the Shapiro-Wilk test, a Pearson (normally distributed) [12] or a Spearman (nonnormally distributed) [12] rank correlation coefficient was used to evaluate the relationship. To evaluate the second hypothesized relationship, a paired t-test (normally distributed) [8] or Wilcoxon signed-rank test (not normally distributed) [8] was employed to determine if there was a statistical difference in grade level between text with disclosure keywords and text without.

3 Results and Analysis

During the experimental phase of this study, specific versions of the previously mentioned Python (v3.13.2) libraries were utilized to compute the variables and results: NLTK (v3.9.3) for tokenization, Textstat (v0.7.13) for readability scoring, and Spacy (v3.8.11) for natural language processing.

3.1 Comparing Disclosure Frequency Score and Policy Readability

To determine whether there was a significant positive relationship between the disclosure score and grade-level readability, we first ran the Shapiro-Wilk Normality Test on both variables (see Table 2). Both were found to be normally distributed ($p > 0.05$). Because of this finding, the Pearson correlation coefficient was used (see Table 3), producing a significant positive relationship ($p < 0.01$) between the disclosure score and grade-level readability, confirming the first hypothesis. This means that as the disclosure score increases, the policy's estimated grade level goes up. Furthermore, the R^2 value of 0.1579 indicates that approximately 15.79% of the variance in a policy's reading grade level can be explained by the disclosure score, which is weak but still significant.

This weak positive correlation is visually represented in the scatter plot in Figure 2, where the x-axis is the Disclosure Frequency Score of the policies, and the y-axis represents the corresponding Flesch-Kincaid Grade Level.

Table 3: Statistical Test Results

Statistical Test	Test Statistic	P-Value
Pearson Correlation	r=0.397	$*1.8235\times10^{-6}$
Wilcoxon Signed-Rank	W=12.000	$*1.2842\times10^{-23}$

* Indicates a significant result

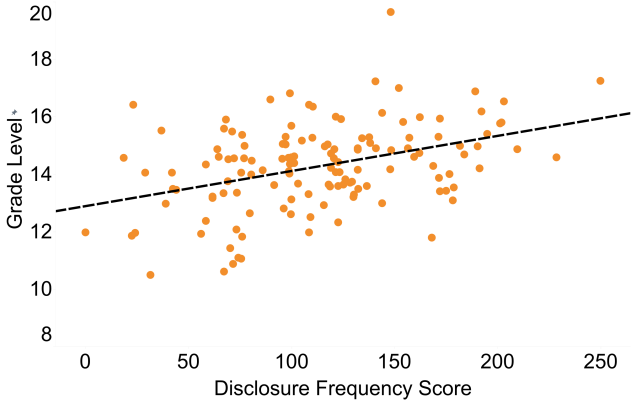

Figure 2: Scatter Plot of Disclosure Frequency Score vs Grade Level

3.2 Comparing Readability Differences Between Policy Sentences

Determining whether there was a significant difference in estimated grade level between sentences with and without disclosure keywords first required running the Shapiro-Wilk Normality Test on the differences across all policies (see Table 2). The differences were not normally distributed ($p < 0.05$). Because the differences were not normally distributed, a Wilcoxon Signed-Rank Test was carried out (see Table 3). The result confirmed the second hypothesis, showing a significant difference ($p < 0.01$) in grade level between sentences containing disclosure keywords and those without. This shows that sentences with disclosure keywords are, on average, harder to read than those without. The median grade level for sentences with keywords was 16.87, while for those without it was around 13.35, or 3.52 grade levels lower.

Figure 3 shows this Grade Level difference. The x-axis shows the calculated grade level, while the rows contain the data points for disclosure sentences and nondisclosure sentences for each policy.

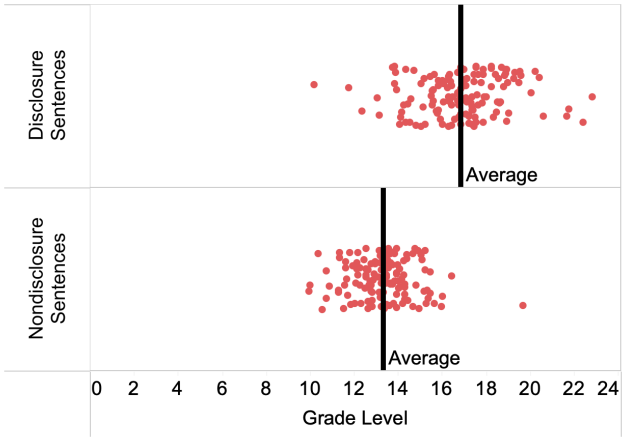

Figure 3: Grade Level Difference Between Disclosure and Nondisclosure Sentences

3.3 Past Work and Limitations

Existing studies have found that the average readability of a privacy policy is significantly worse than that of an average person. This study also confirms that the median grade level is 14.4, about 5 points higher than the recommended grade level for accessibility to the general population. This study extends that finding by also establishing a significant link between disclosures and readability.

The distribution of policy grade levels is shown in Figure 4. In the histogram, the x-axis shows the grade level bins, and the y-axis shows the number of policies in each bin.

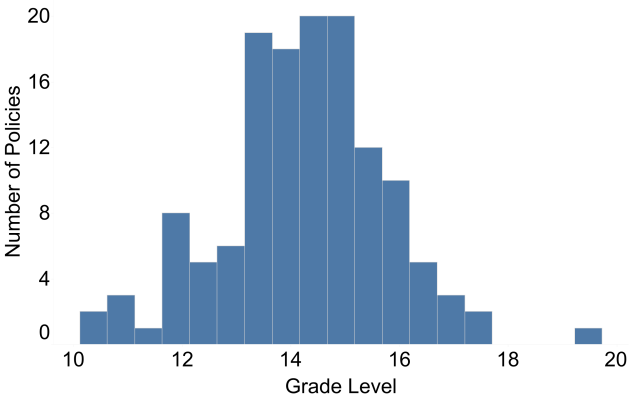

Figure 4: Grade Level Histogram of Policies

This study was limited by the cleaning of the unstructured dataset, which sometimes resulted in the outright elimination of policies due to their structure. If policies were standardized or required to be in a printable format, cleaning would be effortless. This study also utilized a single readability score, but as noted before, the different readability scores in another study on this topic were found to be correlated, so this limitation is minimal. The use of a fixed set of keywords also limited it, because there are endless ways to disclose data handling, and a fixed list could never properly count all of them, nor account for when they are not used as such.

4 Conclusion

The significant link between disclosure keywords and worse readability suggests that privacy policies are harder to understand as more practices are disclosed. While the results were weak, they do not lower their significance. A user who cannot understand a privacy policy cannot possibly be considered informed, which undermines the whole purpose of privacy policies. More concerning is the significant difference in readability between sentences that contain disclosure keywords and those without. The 3.52 grade level increase from sentences without them shows that sentences with the policy's most important details are the hardest to read, which again raises questions about whether current policies properly inform users.

Future work could further minimize limitations by implementing a more thorough cleaning process and including additional readability metrics. Using machine learning to determine where disclosures occur could also improve the disclosure score metric. The differences between industries are another topic considered but not implemented in this study, as is a custom score calculated by large language models to compare with the readability.

References

[1] Andrick Adhikari, Sanchari Das, and Rinku Dewri. 2023. Evolution of Composition, Readability, and Structure of Privacy Policies over Two Decades. *Proceedings on Privacy Enhancing Technologies* 2023, 3 (2023), 138–153. doi:10.56553/popets-2023-0074

[2] Abdulrahman Alabduljabbar, Ahmed Abusnaina, Ülkü Meteriz-Yıldıran, and David Mohaisen. 2021. TLDR: Deep Learning-Based Automated Privacy Policy Annotation with Key Policy Highlights. In *Proceedings of the 20th Workshop on Privacy in the Electronic Society (WPES '21)*. ACM, 103–118. doi:10.1145/3463676.3485608

[3] Bianca Bartelt and Erik Buchmann. 2024. Transparency in Privacy Policies. In *Proceedings of the Twelfth International Conference on Building and Exploring Web Based Environments (WEB 2024)*. 1–7.

[4] Tatiana Ermakova, Benjamin Fabian, and Eleonora Babina. 2015. Readability of Privacy Policies of Healthcare Websites. In *Proceedings of the 12th International Conference on Wirtschaftsinformatik*. Osnabrück, Germany, 1–15.

[5] Benjamin Fabian, Tatiana Ermakova, and Tino Lentz. 2017. Large-Scale Readability Analysis of Privacy Policies. In *Proceedings of the International Conference on Web Intelligence (WI '17)*. ACM, New York, NY, USA, 18–25. doi:10.1145/3106426.3106427

[6] Mark A. Graber, Donna M. D'Alessandro, and Jill Johnson-West. 2002. Reading level of privacy policies on Internet health Web sites. *Journal of Family Practice* 51, 7 (2002), 642–645.

[7] Duha Ibdah, Nada Lachtar, Satya Meenakshi Raparthi, and Anys Bacha. 2021. "Why Should I Read the Privacy Policy, I Just Need the Service": A Study on Attitudes and Perceptions Towards Privacy Policies. *IEEE Access* 9 (2021), 165313–165331. doi:10.1109/ACCESS.2021.3130086

[8] Tae Kyun Kim. 2015. T test as a parametric statistic. *Korean Journal of Anesthesiology* 68, 6 (2015), 540–546. doi:10.4097/kjae.2015.68.6.540

[9] J. Peter Kincaid, Robert P. Fishburne, Richard L. Rogers, and Brad S. Chissom. 1975. *Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel.* Technical Report Research Branch Report 8-75. Naval Technical Training Command, Millington, TN, USA.

[10] Barbara Krumay and Jennifer Klar. 2020. Readability of Privacy Policies. In *34th IFIP Annual Conference on Data and Applications Security and Privacy (DBSec)*. Springer-Verlag, 388–399. doi:10.1007/978-3-030-49669-2_22

[11] Janet Prichard and Kevin Mentzer. 2017. An Analysis of App Privacy Statements. *Issues in Information Systems* 18, 4 (2017), 179–188. doi:10.48009/4_iis_2017_179-

188

[12] Andrijana Rebekić, Zdenko Lončarić, Sonja Petrović, and Sonja Marić. 2015. Pearson's or Spearman's Correlation Coefficient - Which One to Use? *Poljoprivreda* 21, 2 (2015), 47–54. doi:10.18047/poljo.21.2.8

[13] Norman Sadeh, Alessandro Acquisti, Travis D. Breaux, Lorrie Faith Cranor, Aleecia M. McDonald, Joel R. Reidenberg, Noah A. Smith, Fei Liu, N. Cameron Russell, Florian Schaub, and Shomir Wilson. 2013. *The Usable Privacy Policy Project: Combining Crowdsourcing, Machine Learning and Natural Language Processing to Semi-Automatically Answer Those Privacy Questions Users Care About.* Technical Report CMU-ISR-13-119. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

[14] Charlotte A. Tschider. 2021. Meaningful Choice: A History of Consent and Alternatives to the Consent Myth. *North Carolina Journal of Law Technology* 22, 4 (2021), 617–680. https://scholarship.law.unc.edu/ncjolt/vol22/iss4/3

[15] Isabel Wagner. 2022. Privacy Policies Across the Ages: Content and Readability of Privacy Policies 1996–2021. *arXiv preprint arXiv:2201.08739* (2022). https://arxiv.org/abs/2201.08739

[16] Tiffany M. Walsh and Teresa A. Volsko. 2008. Readability Assessment of Internet-Based Consumer Health Information. *Respiratory Care* 53, 12 (2008), 1710–1715.

Exploring Boolean Dependency Satisfaction for Rust Crates Aimed at Embedded Systems

Hamilton Ferris
hamiltonaferris@gmail.com
Winona State University
Winona, Minnesota, USA

Abstract

Rust crates are concise, safe ways to abstract compilation units in the Rust programming language. Each crate can be compiled individually, or some may depend on others (i.e. function 'bar' may depend on crate 'foo'). However, there is no standard way to compile Rust crates in a non-standard way, which targets the compilation at embedded systems.

Existing research into this topic mostly focuses on developer opinions. Sharma et al. discovered that developers at and around October 2024 had mixed opinions about the state of Rust and embedded systems, with only 68% of developers thinking Rust crate support was sufficient. In addition, 92% of developers surveyed also used C for embedded systems, and 64% of which claimed an improvement in code safety after switching to Rust. Overall, Rust could be more widely adopted and improved even further with easier ways to correctly evaluate compiler options.

This paper tests the open problem P0.1 in Sharma et al., which hypothesizes the idea of compiling a given Rust crate and its dependencies using a binary constraint satisfaction algorithm to derive the correct compilation flags. The results, while limited, suggest that this approach is not effective in deriving correct compiler feature flags in the current Rust ecosystem.

CCS Concepts

• **Computer systems organization** → *Embedded software*; • **Software and its engineering** → *Software usability*.

Keywords

Embedded Systems, Rust feature flags, Cargo, Binary Constraint Satisfaction Algorithms

ACM Reference Format:
Hamilton Ferris. 2026. Exploring Boolean Dependency Satisfaction for Rust Crates Aimed at Embedded Systems. In *Proceedings of the 26th Winona Computer Science Undergraduate Research Seminar*. ACM, New York, NY, USA, 4 pages.

1 Introduction

The Rust programming language contains many features meant to help software developers with low-level problems that are common in C such as type safety, use-after-frees, and in some cases unhelpful compiler errors. In October of 2024, an annual paper detailing the current state of Rust among software developers was released [7] and included was a survey of embedded systems developers focused on Rust in embedded systems. In the paper, the authors find that existing developer support for Rust is mixed. Only 68% of developers they studied answered that they found the Rust crates support for embedded systems development sufficient. This conclusion is furthered by two additional factors that they must consider: larger overhead and Rust-specific bugs [4]. Another paper showed that typically, more specialized programs utilizing Rust-specific features take longer to compile [8]. In addition, another paper found that the same kinds of Rust-specific compiler bugs can affect embedded systems adoption including ones that can affect the compilation outcomes of a developer compiling and optimizing for an embedded system [3].

In the paper authored by Sharma et al., they brainstormed the current open problem P0.1, which hypothesizes that creating a binary constraint satisfaction algorithm to improve reliable non-standard compiling in Rust would be a better way to get embedded systems developers to adopt it. Binary constraint satisfaction problems are ways of mapping the correctness of a variable that can only be one of two options. In this instance, a compiler flag can either be enabled or disabled. Nemhauser and Wolsey authored a big foundational piece of developing a model like this by exampling integer programming and integer lattices [5]. A model developed using these principles with modern approaches can be tested on Cargo crates so that compiling crates for embedded systems may be more reliable and perhaps, even faster.

Embedded projects created in Rust utilize special crate configurations that are known as no_std, of which the Rust compiler compiles them, making no assumptions about the hardware it's running on [2]. As such, a crate marked with no_std has no API for system-specific utilities or platform integration, and lacks the runtime provided by Rust's `libstd` [2]. This paper aims to address the issues that may arise when compiling a given crate that supports no_std, where crates may have different flags, options, or configurations to successfully compile using no_std.

2 Background Research

Existing research into binary constraint satisfaction algorithms is quite extensive, mostly covering consistent improvements to algorithms [1], [6], but doesn't usually venture into the realm of compiler flags. Research about Rust's developer opinion is fairly extensive, mostly covering broad features for lots of developers, since Rust is still an up-and-coming programming language. Problem P0.1 from Sharma et al. supposes that converting compiler feature flags to a binary constraint object model, then solving that using a satisfaction algorithm could be a cleaner way to package crates in Rust, specifically those that target embedded systems.

Given this information, this study hypothesizes that a binary constraint satisfaction algorithm will derive a minimal set of feature flags for successful compilation of a given no_std Rust crate.

the 26th Winona Computer Science Undergraduate Research Seminar, Winona, MN, US 2026.

3 Methodology

First, we define some terms and ideas used in this paper.

A crate is considered `no_std` if it contains a feature or function that disables the standard library, such as declaring `#![no_std]` or `#![cfg_attr(not(feature = "std"), no_std)]` in the beginning of a `Rust` file. The former is an explicit declaration of a non-standard `Rust` file, while the latter declares that if a `Rust` file is compiled without standard features, switch to `no_std`, rather than simply compiling without the standard library. This is most common in crates that support both `std` and `no_std`.

"Successful compilation" is defined as the `Rust` package manager, `Cargo`, throwing no error when attempting to check or compile a source file. Warnings are allowed, but any step that halts compilation early or fails to produce an output file is considered a failed compilation.

A feature flag, in `Rust`, is a feature that can be optionally included or excluded at compile time. For non-standard `Rust` code, these features are sometimes required for `Cargo` to successfully check a given program against `no_std`. This is because when –no-default-features is enabled, `Cargo` does not use its internal build system which normally resolves crate dependencies for the user.

Finally, the target `thumbv7-none-eabi` was chosen as the `no_std` target because it is one of the most widely accepted `no_std` `Rust` targets for developers. It defines a target for a class of ARM Cortex microcontrollers, namely the M3, M4, and M7 chips. 'thumbv7' defines the instruction set (ARM Thumb Version 7), 'none' defines the operating system (of which there is none), and 'eabi' stands for Embedded Application Binary Interface, which defines some key information about how variables are passed into certain registers.

`Rust` crates were sorted by crates that support `no_std` according to a third-party website that supports crate filtering, lib.rs. The crates were selected according to number of downloads, total dependencies, and must have been marked as `no_std`. Crates were hand-picked to select a diverse group of crates that were able to be hand-compiled to test correctness. Ten crates were selected, demonstrated in Table 1. Some crates were selected to evaluate crate-specific constraints over general ones, like prost, which implements prost-derive and derive. Others, like nalgebra and num-traits, were selected as a baseline to evaluate the correctness of the binary constraint satisfaction algorithm.

Table 1: Crates, versions, required feature flags, and monthly downloads (April 2026)

Name	Version	Required Features	Downloads
heapless	0.9.2	None	8,198,284
embedded-hal	1.0.0	None	1,325,552
funty	3.0.0-rc2	None	11,623,208
micromath	2.1.0	None	122,658
num-traits	0.2.19	libm	40,111,275
nalgebra	0.34.2	libm, alloc	4,329,523
serde	1.0.228	alloc, derive	55,568,821
hashbrown	0.16.1	alloc, allocator-api2	125,929,757
bitvec	1.0.1	alloc, atomic	11,970,691
prost	0.14.3	alloc, prost_derive	30,828,671

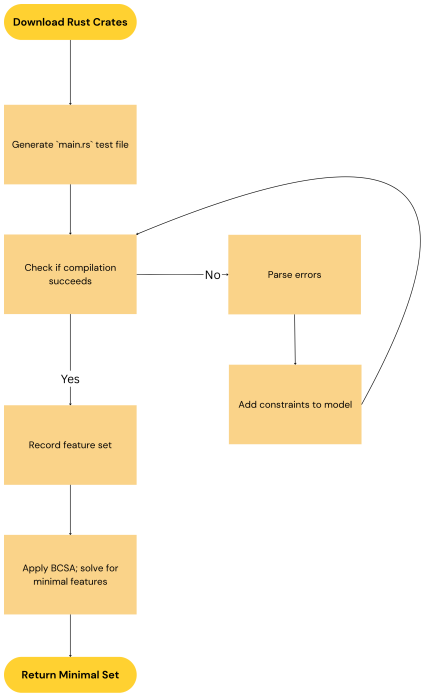

Figure 1: Algorithm Flow Chart

Each of these crates has some combination of feature flags required to compile properly to `no_std`. This can be shown as the following:

Let $F = \{f_1, f_2, \ldots, f_n\}$ be the set of feature flags for a crate. Each crate C has a subset $F_C \subseteq F$, which contains the minimum set of feature flags required to compile under `no_std`.

The binary constraint satisfaction algorithm (BCSA) is built using the Python library PuLP, using the HiGHS solver. Previous iterations of the program used Coin or Branch and Cut (CBC), and moved to HiGHS for better performance, though they both performed the same when comparing correctness.

Crates were checked for correctness by running the command:

```
cargo check –no-default-features
–target=thumbv7-none-eabi.
```

Running `Cargo` using the check feature allows `Cargo` to test compilation under a set of restrictions, like using disabling the standard library (`–no-default-features`) and specifying crates to link and a target to compile for.

To perform this experiment, a Python script was created that downloads the crates from crates.io. For each crate in the list, a `main.rs` script was created to expose the crate's functions. This `main.rs` was placed in a test directory with the crate added, then the check command was called. Any standard error that was generated was parsed into known feature flags (i.e. "`Vec` not found…" → `alloc`).

4 Results & Analysis

This experiment was conducted using `Cargo` and `rustc` version 1.94.1, as well as `Python` 3.14.4 and `Poetry` 2.3.4.

Table 2: Expected features vs. derived features & compilation outcomes

Crate	Expected minimal feature set	BCSA derived set	Compilation outcome
heapless	[]	[]	Success
embedded-hal	[]	[]	Success
funty	[]	[]	Success
micromath	[]	[]	Success
num-traits	[libm]	[libm]	Success
nalgebra	[libm, alloc]	[alloc]	Fail, needs libm
serde	[alloc, derive]	[alloc, derive]	Success
hashbrown	[alloc, default-hasher]	[alloc, libm, default-hasher]	Fail, remove libm
bitvec	[alloc, atomic]	[alloc]	Success
prost	[prost-derive, alloc]	[derive]	Success

Each crate was hand-selected to provide the best combination of manual feature flag testing and complexity for the BCSA. There are four crates that have no feature flag dependencies (`heapless`, `embedded-hal`, `funty`, and `micromath`), one that has one dependency (`num-traits` depends on `libm`), and five crates that have two dependencies (`nalgebra`, `serde`, `hashbrown`, `bitvec`, and `prost`). `micromath` in particular was chosen because it implements features from `libm`, but does not depend on it. This requires the feature parser to correctly identify that `num-traits` requires `libm`, but `micromath`, using the same methods, requires its own implementation.

After running the Python script, any feature flag detected by the BCSA was placed into the `Cargo.toml` under a dependency header, then the compilation check was re-run. To allow the BCSA to correctly detect all feature flags, all crates were compiled prior to running the algorithm. Error messages from the standard error were placed into a Python file, which matched certain messages with associated features (i.e. Box -> alloc, Float -> libm, etc.). In addition, a `main.rs` file was placed in the test's `src/` directory to expose the crate's API and get a true positive or negative for missing or non-missing feature flags. This `main.rs` was created using ChatGPT free tier, which uses the GPT-5.2 mini model. ChatGPT was prompted with a crate's functions provided and asked to create a non-standard compliant `main.rs` file that uses all the features of a given crate, then double-checked for correctness by hand.

The BCSA identifies a set of feature flags for all crates, though the set is not always correct. During compilation testing, only eight of ten went through the check without any errors. As shown in table 2, we see that the set of feature flags the BCSA derives is correct 73% of the time (11 flags derive / 15 flags to derive). All four of the crates with empty feature flag sets (aka no feature flags required) were 100% accurate, the BCSA correctly derived `libm` for `num-traits`, but when prompted for two or more flags the feature flag derivation accuracy falls to 60%, and actually incorrectly assumes `hashbrown` needs `libm`.

An interesting observation is that `bitvec` and `prost` both succeeded, despite the BCSA set not matching the expected set. This is most likely an error in usage and input. In the case of `bitvec`, `atomic` exists to alter the types that the code uses (u8 -> AtomicU8). In this case, the BCSA derived set produces a successful compilation, but does not match the expected feature set. `prost`, on the other hand, uses `prost-derive` to encode and decode APIs. In this case, `prost-derive` provides a much nicer implementation of `prost`'s API features than the default — which would be a real hurdle that someone using this algorithm would have to overcome and therefore the output produced is actually wrong, and so the BCSA produces a false positive. This means that out of the 10 total compilations, only 6 of them produced a true positive compilation. Both `nalgebra` and `hashbrown` had extra or missing features, `bitvec` did not detect the atomic flag, and `prost` produced a false positive.

5 Conclusion

In conclusion, the results from this study provide limited evidence supporting the hypothesis; that a binary constraint satisfaction algorithm is a good way to derive Cargo feature flags for `Rust` crates. Compared to just examining the `Cargo` features provided, the BCSA provides unnecessary overhead, time to compile, and isn't completely accurate. It also requires a vast amount of prerequisite mapping between errors and their feature flags, of which both could change over time. The algorithm should not be put out entirely, as it could work if the `Rust` environment changes drastically, or for specific environments conducive to large-overhead, safe options like this BCSA.

5.1 Limitations & Future Research

The biggest limitation of the current research is that there exists no scope at which this current problem might be implemented smoothly and correctly given the overhead required to make it function. Modeling a given feature flag as a binary constraint problem is certainly more efficient than a brute-force solution, though the differences in feature flag error detection are both verbose and heavy. Perhaps future research could be conducted on a broader scale, where either the overhead makes sense, or the environment dynamically detects error to feature flag mapping, rather than hard coding it in ahead of time. Some additional cost analysis on flags that do not error out but add bloat (compile time, binary size, etc) could be attempted or even a different algorithm entirely.

One smaller but important limitation is the physical implementation of an algorithm like this one. Questions like "Where in `Cargo` might this be implemented?" and "What kinds of features should we target?" should be asked before moving forward with a BCSA-centric approach. Other things to consider would be maintaining the features, updating standard output and error parsing, and more.

An alternative to brute-forcing the BCSA input could be to standardize `Cargo` input by forcing any crate dependencies to be declared under a different, `no_std` header in `Cargo.toml`. The biggest upside of this approach is that it fully gets rid of one of the biggest hurdles, but the obvious downside is that to succeed, developers would need to collectively agree to a new `Rust` standard.

References

[1] Fahiem Bacchus, Xinguo Chen, Peter Van Beek, and Toby Walsh. 2002. Binary vs. Non-Binary Constraints. *Artificial Intelligence* 140, 1-2 (2002), 1–37. doi:10.1016/S0004-3702(02)00210-2

[2] Rust Embedded. 2025. The Embedded Rust Book. GitHub. https://docs.rust-embedded.org/book/

[3] Ranjit Jhala and Rupak Majumdar. 2009. Software Checking. *Comput. Surveys* 41, 4 (2009), 1–54. https://dl.acm.org/doi/epdf/10.1145/1592434.1592438

[4] Zhengyu Liu, Yifan Feng, Yuhang Ni, Shuang Li, Qirun Shi, Bo Xu, and Zhendong Su. 2025. An Empirical Study of Rust-Specific Bugs in the rustc Compiler. *Association for Computing Machinery* 9, 411 (2025), 1–28. doi:10.1145/3763800

[5] George Nemhauser and Laurence Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc. https://onlinelibrary.wiley.com/doi/book/10.1002/9781118627372

[6] Pablo San Segundo, Fabio Furini, and Rafael León. 2021. A new branch-and-filter exact algorithm for binary constraint satisfaction problems. *European Journal of Operational Research* 299, 2 (2021), 448–467. doi:10.1016/j.ejor.2021.09.014

[7] Aman Sharma, Shubham Sharma, S. R. Tanksalkar, Santiago Torres-Arias, and Aravind Machiry. 2024. Rust for Embedded Systems: Current State and Open Problems. In *Proceedings of the ACM CCS 2024*. 2296–2310. doi:10.1145/3658644.3690275

[8] Amir Syalim and D. P. Sheradhien. 2025. C vs Rust: Manual vs Automatic Spacial and Temporal Memory Safety. *The Indonesian Journal of Computer Science* 14, 2 (2025), 2197–2213. http://www.ijcs.net/ijcs/index.php/ijcs/article/view/4640/1006

Comparative Prediction of Soccer Match Result with Machine Learning Models

Pronob Kumar*
Winona State University
Winona, Minnesota, USA

Abstract

Predicting soccer match outcomes is a challenging problem due to the sport's low-scoring nature and the high frequency of draw results. This study uses a historical dataset of European league matches from 1960 to 2024 to develop and evaluate machine learning models for predicting match outcomes. Several models, including logistic regression, decision trees, random forests, and XGBoost, were implemented and compared.

Initial models using basic features achieved moderate performance, with XGBoost reaching approximately 65% accuracy. To improve predictions, team-level rolling performance features and relative difference metrics were introduced to capture recent team strength. The enhanced model achieved improved performance, reaching approximately 67% accuracy.

Despite these improvements, all models consistently struggled to predict draw outcomes, indicating the inherent uncertainty associated with such results. Overall, the results demonstrate that while machine learning models can effectively capture dominant patterns in soccer matches, predicting less frequent outcomes remains a significant challenge and an important direction for future research.

Keywords

machine learning, soccer prediction, classification, XGBoost, random forest

1 Introduction

Soccer is one of the most unpredictable games in the world. This game is also considered a low-scoring game compared with many other sports. Across many leagues and competitions, the average total goals per match typically ranges between approximately 2.4 to 2.7 goals per game [4]. On top of that, it also has a high rate of draw outcomes, which makes prediction harder. Based on recent statistics, about 23% of matches end in a draw, roughly one in four matches [5]. It also has factors like player injuries and ground variance. Due to this unpredictability, various bets are placed every year on it. Among all these bets, most of them are not calculated, which leads to frequent losses. To address this issue, machine learning can be used to predict match results.

The problem of predicting soccer match results has attracted increasing attention from researchers due to the sport's inherent uncertainty and the availability of large historical datasets. Traditional statistical approaches, such as Poisson and regression-based models, have been widely used to model goal scoring and match outcomes by assuming low event rates and independence between scoring events [2]. More recently, machine learning techniques have been applied to this problem to capture nonlinear relationships and interactions among multiple match-related factors [6]. Prior studies have explored methods such as logistic regression, decision trees, random forests, and gradient boosting to predict match outcomes or goal counts, often demonstrating improvements over simple baseline models [2]. However, despite these advancements, accurately predicting draw outcomes remains difficult.

In addition to machine learning approaches, rating-based systems such as ELO have also been used to estimate team strength and predict match outcomes [7]. These approaches provide a dynamic way to track team performance over time but may not fully capture short-term variations in team form.

In this study, soccer match result prediction is formulated as a supervised machine learning classification problem, where historical match data are used to predict future outcomes. The proposed approach focuses on modeling match results using machine learning algorithms while incorporating team-level performance indicators. The contribution of this work lies in systematically analyzing how these features affect predictive performance and comparing multiple machine learning models under a consistent evaluation framework.

1.1 Hypothesis

The machine learning model, such as decision trees and random forests, will be able to predict match results 2% more accurately than the baseline model, such as a regression model, on historical European league data.

1.2 Research Question

Can machine learning models predict soccer match outcomes more accurately than simple baseline models based on historical outcome data?

2 Background Research

Predicting soccer match outcomes has been a long-standing problem in sports analytics due to the inherent uncertainty and low-scoring nature of the game. Early approaches to this problem relied heavily on statistical models, particularly Poisson-based methods, which model the number of goals scored by each team as independent random variables. These models are well-suited for soccer because goals occur infrequently; however, they rely on strong assumptions such as independence between scoring events and constant scoring rates, which often do not hold in real-world matches [2].

*Advised by Dr. Mingrui Zhang and Dr. Trung Nguyen

Proceedings of the 26th Winona Computer Science Undergraduate Research Seminar, Winona, MN, US
2026.

To address these limitations, researchers have explored probabilistic rating systems such as the ELO rating model, which dynamically updates team strength based on past performance and match outcomes. These models provide a more flexible representation of team ability over time and have been shown to improve predictive performance compared to static statistical approaches [7]. However, while ELO-based methods capture long-term team strength effectively, they may fail to reflect short-term fluctuations in team performance, such as recent form or temporary changes in lineup.

With the advancement of data availability and computational power, machine learning methods have become increasingly prominent in soccer prediction tasks. Logistic regression is often used as a baseline model due to its simplicity and interpretability [8]. Decision trees extend this approach by learning hierarchical decision rules that can capture nonlinear relationships between variables [11]. Random forests improve upon decision trees by combining multiple trees to reduce variance and improve generalization [1]. More recently, gradient boosting methods, particularly XGBoost, have gained popularity due to their ability to iteratively refine predictions and handle complex interactions within structured data [3].

Several studies have demonstrated that ensemble methods such as random forests and gradient boosting tend to outperform simpler models in predicting match outcomes [6, 12]. In addition, recent research emphasizes the importance of incorporating dynamic features that capture team performance over time. Features such as rolling averages of goals, win rates, and goal differences provide a more realistic representation of team strength compared to static aggregate statistics [12]. These approaches allow models to adapt to changes in team performance and improve predictive accuracy.

Despite these advancements, predicting draw outcomes remains a persistent challenge. Draws occur less frequently and often result from closely matched teams or unpredictable match conditions, making them difficult to model accurately. This limitation highlights the need for more sophisticated feature engineering and modeling techniques that can better capture uncertainty and balance between competing teams.

3 Methods: Research

The dataset used in this study was obtained from Kaggle and contains historical European soccer matches from 1960 to 2024 [10]. The data includes information such as team identities, match results, goals scored, match dates, and stadium-related variables.

Prior to analysis, the dataset was cleaned by removing irrelevant attributes and handling missing values to ensure consistency and reliability. Columns with excessive missing data or those that could introduce data leakage, such as post-match statistics, were excluded. Remaining missing values in numerical features were handled using appropriate imputation techniques.

Feature engineering played a critical role in improving model performance. Instead of relying solely on static features such as team names, dynamic team-level performance indicators were constructed. These indicators were based on rolling statistics computed from historical match data. For each team, rolling averages of goals scored, goals conceded, and win rates were calculated using a fixed window of the previous matches. This approach allows the model to capture recent form and performance trends, which are essential in sports analytics [12].

To ensure the validity of the predictive framework, all rolling features were constructed using only information available prior to each match. This was achieved by shifting the data before computing rolling averages, thereby preventing data leakage and ensuring that the model does not have access to future information during training. Such practices are critical in time-dependent prediction problems [9].

In addition to individual team performance metrics, relative difference features were introduced to capture the comparative strength between competing teams. These features include differences in win rates, goal-scoring averages, and defensive performance between the home and away teams. Prior research suggests that relative performance measures are often more informative than absolute metrics when comparing competing entities [11]. Furthermore, derived features such as goal balance were included to provide a more comprehensive representation of team strength.

The predictive models used in this study include logistic regression, decision tree, random forest, and XGBoost. Logistic regression serves as a baseline model and provides a probabilistic interpretation of classification outcomes [8]. Decision trees model the data through hierarchical splits, allowing for nonlinear relationships [11]. Random forests extend this approach by combining multiple decision trees to improve robustness and reduce overfitting [1]. XGBoost, a gradient boosting algorithm, builds models sequentially by minimizing prediction errors and has demonstrated strong performance in structured data problems [3].

To improve model performance, hyperparameter tuning was performed, particularly for tree-based and boosting models. Parameters such as the number of estimators, maximum tree depth, and learning rate were adjusted to balance bias and variance. For XGBoost, additional parameters such as subsampling rates and column sampling were also considered to improve generalization.

The dataset was divided into training and testing sets using a chronological split, where 80% of the data was used for training and the remaining 20% for testing. This approach ensures that models are evaluated on future data, simulating real-world prediction scenarios and preventing data leakage.

Model performance was evaluated using multiple metrics, including accuracy, precision, recall, and F1-score. Confusion matrices were used to provide a detailed breakdown of classification performance across outcome classes, namely home win (H), draw (D), and away win (A). Receiver Operating Characteristic (ROC) curves were also generated to evaluate the models' ability to distinguish between classes under different thresholds.

4 Results and Analysis

After preprocessing and feature engineering, the dataset consisted of approximately 2,372 observations and 26 features capturing both team performance and match context. The engineered features included rolling performance metrics, such as recent win rates and goal statistics, as well as relative difference features representing the comparative strength between teams. The distribution of match outcomes revealed a class imbalance, with home wins occurring most frequently, followed by away wins, while draw outcomes

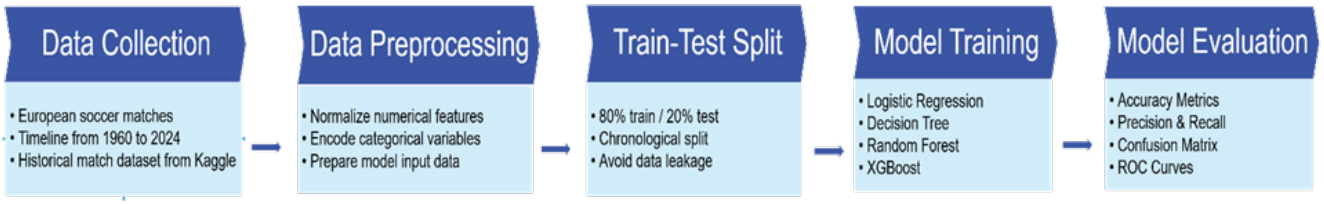

Figure 1: Overview of the machine learning pipeline, including data collection, preprocessing, feature engineering, model training, and evaluation.

represented the smallest proportion of the dataset. This imbalance is consistent with known characteristics of soccer data and has been noted as a challenge in prior studies [5, 12].

All experiments were conducted using Python (version 3.11), with libraries including Pandas (version 2.0) for data preprocessing, Scikit-learn (version 1.3) for model implementation, and XGBoost (version 1.7) for gradient boosting. The models were implemented using standard machine learning pipelines, and hyperparameter tuning was performed for tree-based models by adjusting parameters such as the number of estimators, maximum tree depth, and learning rate. The dataset was divided using a chronological split, where 80% of the data was used for training and 20% for testing. This approach ensures that models are evaluated on future data, which is critical in time-dependent prediction problems and helps prevent data leakage [9].

To evaluate model performance, multiple experiments were conducted comparing baseline models with progressively enhanced feature sets. The results of these experiments are summarized in Table 1.

Table 1: Model Comparison Results

Model	Accuracy	H Recall	A Recall	D Recall
Logistic Regression	0.50	0.55	0.50	0.10
Decision Tree	0.51	0.48	0.70	0.21
Random Forest	0.59	0.69	0.67	0.12
XGBoost	0.65	0.81	0.70	0.10

As shown in Table 1, XGBoost achieved the highest performance among the baseline models, with an accuracy of approximately 65%, outperforming logistic regression, decision trees, and random forests. These results are consistent with previous research demonstrating the effectiveness of gradient boosting methods in structured data problems [3]. When team-level rolling features were introduced, the overall accuracy did not significantly improve. However, the inclusion of relative difference features led to a measurable improvement, with the final XGBoost model achieving approximately 67% accuracy. This result supports the hypothesis that relative team performance provides more predictive value than absolute metrics.

Further insight into model performance is provided by the confusion matrices shown in Figure 2.

Figure 2 illustrates the classification results for each model, where the rows represent the actual outcomes and the columns represent predicted outcomes. The labels H, D, and A correspond

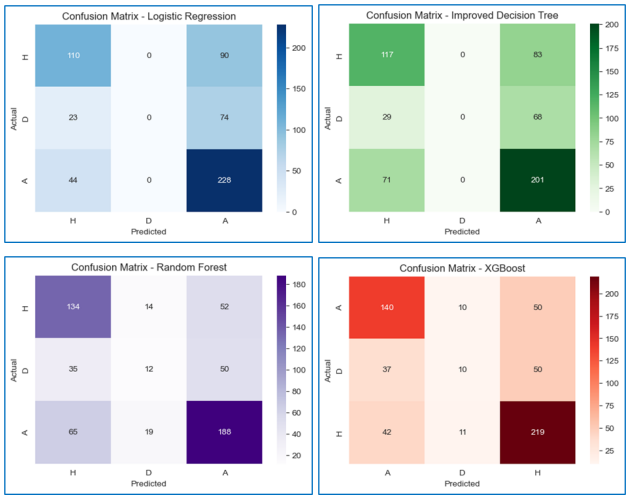

Figure 2: Confusion matrices comparing model performance. Rows represent actual outcomes and columns represent predicted outcomes. H, D, and A correspond to home win, draw, and away win.

to Home Win, Draw, and Away Win, respectively. The confusion matrices show that the models perform well in predicting home and away wins, as indicated by the higher values along the diagonal for these classes. However, draw outcomes are frequently misclassified as either home or away wins. This behavior suggests that the model struggles to identify balanced match conditions, which has been observed in prior studies on soccer prediction [12].

To further evaluate model performance, Receiver Operating Characteristic (ROC) curves were generated for selected classification scenarios, as shown in Figures 3 and 4.

Figures 3 and 4 show that the models achieve stronger discriminative performance for home win predictions compared to draw predictions. The ROC curve for home wins demonstrates a higher true positive rate across different thresholds, while the curve for draw predictions remains significantly lower. This indicates that the model is less capable of distinguishing draw outcomes from other classes, reflecting the inherent uncertainty associated with such results.

Overall, these results suggest that while machine learning models can effectively capture dominant patterns in soccer match outcomes, predicting draw results remains a significant challenge. The improvement achieved through feature engineering highlights the

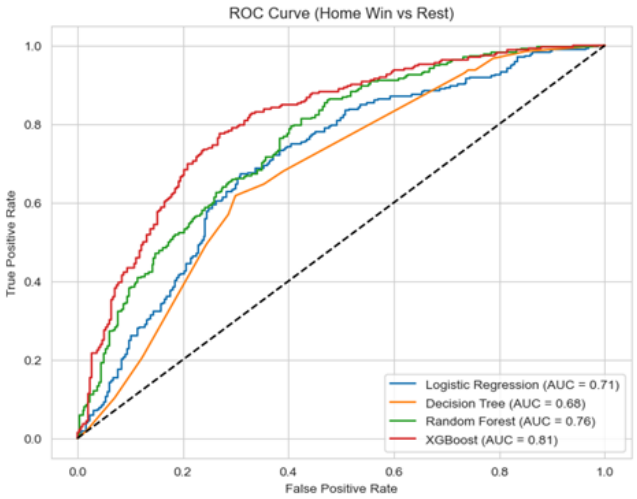

Figure 3: ROC curve for Home Win versus Rest classification.

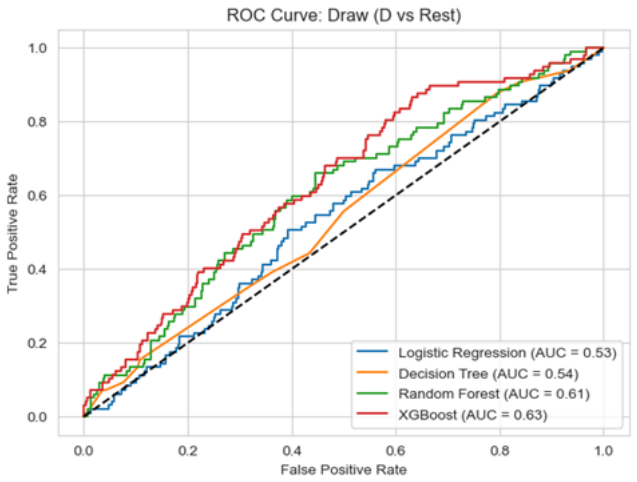

Figure 4: ROC curve for Draw versus Rest classification.

importance of incorporating relative team strength, but also indicates that additional features or alternative modeling approaches may be required to further enhance predictive performance.

5 Conclusion

This study investigated the effectiveness of machine learning models in predicting soccer match outcomes using a historical dataset spanning multiple decades. By implementing and systematically comparing several models, including logistic regression, decision trees, random forests, and XGBoost, this research provided a comprehensive evaluation of different predictive approaches for a complex and inherently uncertain problem. The use of multiple models allowed for a balanced comparison between simple, interpretable methods and more advanced ensemble techniques, which have been widely recognized for their effectiveness in structured data applications [3, 8].

A central contribution of this work lies in the design and incorporation of team-level feature engineering techniques. In particular, rolling performance metrics and relative difference features were introduced to capture both recent team form and comparative strength between competing teams. These features provided a more realistic representation of match dynamics compared to traditional static features. The results demonstrated that while basic rolling features alone did not significantly improve performance, the inclusion of relative difference metrics led to a measurable improvement in predictive accuracy. The final XGBoost model achieved an accuracy of approximately 67%, representing an improvement of about 2% over baseline approaches. This finding supports the hypothesis that enhanced feature engineering can improve model performance and aligns with prior research emphasizing the importance of dynamic and relative features in sports prediction tasks [11, 12].

Despite these improvements, the study also revealed a significant limitation in predicting draw outcomes. Across all models, the recall for draw predictions remained substantially lower than for home and away wins. This result highlights the inherent difficulty of modeling draw outcomes, which often arise from closely matched teams and unpredictable match conditions. Similar challenges have been reported in previous studies, where draw predictions consistently exhibit lower performance due to class imbalance and higher uncertainty [5, 12]. The imbalance in class distribution may negatively affect the model's ability to learn patterns associated with draw outcomes, as fewer training examples are available for this class.

Future work could explore techniques specifically designed to address class imbalance, such as the Synthetic Minority Over-sampling Technique (SMOTE), which generates additional synthetic samples for minority classes based on existing data. Incorporating such methods may improve the model's ability to better capture draw outcomes and enhance overall predictive performance. In addition, exploring cost-sensitive learning or class-weighted models may further help address this imbalance and improve classification performance for underrepresented outcomes.

Another important contribution of this study is the methodological framework used to ensure valid model evaluation. By employing a chronological train-test split, the study simulated real-world prediction scenarios and avoided data leakage, which is a critical consideration in time-dependent datasets [9]. In addition, hyperparameter tuning was applied to optimize model performance, particularly for ensemble methods such as random forests and XGBoost. These methodological choices strengthen the reliability of the results and provide a robust foundation for future research.

Looking forward, there are several avenues for improving predictive performance in soccer match outcome modeling. One potential direction is the incorporation of more granular data, such as player-level statistics, team lineups, and injury reports, which could provide deeper insights into match dynamics. Another promising approach is the integration of external data sources, such as betting odds or expert predictions, which have been shown to contain valuable information about expected match outcomes. Additionally, advanced modeling techniques, including deep learning architectures and hybrid ensemble models, could be explored to better capture complex relationships within the data.

In conclusion, this study demonstrates that machine learning models can effectively capture dominant patterns in soccer match outcomes, particularly for home and away wins. However, predicting draw outcomes remains a significant challenge, highlighting both the limitations of current modeling approaches and the inherent uncertainty of the sport. The findings of this research contribute to a deeper understanding of soccer prediction and provide a foundation for future work aimed at improving model performance through enhanced feature engineering and data integration.

References

[1] Leo Breiman. 2001. Random Forests. *Machine Learning* (2001).

[2] R. Bunker, C. Yeung, and K. Fujii. 2024. Machine Learning for Soccer Match Result Prediction. arXiv. doi:10.48550/arXiv.2403.07669

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[4] Dan. 2016. Most soccer matches are within one goal of 1-0. https://www.r-bloggers.com/2016/03/most-soccer-matches-are-within-one-goal-of-1-0/. Accessed 2026.

[5] FootyStats. n.d.. Football Draws Stats. https://footystats.org/stats/draws. Accessed 2026.

[6] Usman Haruna, Jaafar Zubairu Maitama, M. Mohammed, and Ram Gopal Raj. 2022. Predicting the Outcomes of Football Matches Using Machine Learning Approach. In *Communications in Computer and Information Science*. 92–104. doi:10.1007/978-3-030-95630-1_7

[7] Lars Magnus Hvattum and Halvard Arntzen. 2010. Using ELO Ratings for Match Result Prediction in Association Football. *International Journal of Forecasting* (2010).

[8] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Springer.

[9] Max Kuhn and Kjell Johnson. 2013. *Applied Predictive Modeling*. Springer.

[10] Piterfm. n.d.. Football / Soccer UEFA Euro 1960–2024 Dataset. https://www.kaggle.com/datasets/piterfm/football-soccer-uefa-euro-1960-2024?resource=download. Kaggle dataset. Accessed 2026.

[11] J. R. Quinlan. 1986. Induction of Decision Trees. *Machine Learning* (1986).

[12] Niek Tax, Yme Joustra, and Wil van der Aalst. 2015. Predicting the Dutch Football Competition Using Public Data. In *Machine Learning and Data Mining in Pattern Recognition*.

Impact of News Sentiment Analysis on LSTM Based Stock Trading Model Performance

Trevor Nindl
Winona State University
Department of Computer Science
Winona, Minnesota, USA

Abstract

This study investigates whether incorporating FinBERT based news sentiment analysis into a Long Short-Term Memory (LSTM) neural network improves its risk adjusted intraday trading performance. Two models were trained on five-minute bar data for seven different stocks that spanned from 2018–2026. The baseline model used 24 technical features whereas the sentiment enhanced model had 25, with FinBERT score being the only difference. Walk-forward validation across five folds was used to prevent look ahead bias when testing the models. Results showed that sentiment improved the Sharpe ratio on four of the seven tickers, with the strongest gains being high volatility, news driven stocks (+3.26 TSLA & +2.27 AAPL), and the negative gains on macro driven stocks (-2.80 JNJ & -1.21 SPY), which suggest that FinBERT inclusion thrives on individual company news that has high price impact.

Keywords

LSTM, FinBERT, intraday trading, sentiment analysis, stock trading, walk forward validation

1 Introduction

Stock price predictions have remained one of the most complex problems in the financial field. This is primarily due to market randomness and the many human factors that influence price fluctuations. Things like the Efficient Market Hypothesis believe that stock prices follow a random walk pattern that makes their movements unpredictable based purely on historical data. This randomness is then further amplified by events such as COVID-19, which produce abnormal market behavior. Advances in neural network models have given us new approaches to combat this randomness. Long Short-Term Memory networks are particularly well suited for this use because their memory cells can retain temporal dependencies across long sequences, allowing the models to draw conclusions on hours of prior data while also responding to the short-term dynamics [7].

Stock price prediction methods have changed and evolved in correlation with technology over the past few decades. Early approaches used things like statistical models that assumed relationships in price data were linear. These struggled with non-stationary behavior of financial markets that newer methods do not. Eventually, machine learning would improve these models by capturing the nonlinear patterns in historical price data. One of the most recent advances in methods used is deep learning architectures such as LSTM networks. Strong results using these networks have been shown through running tests on sequential financial data [7]. These advances shifted the methods being used from rule-based trading to data driven models that are now capable of adapting to changes seen in market data. The next advancement on the rise is the inclusion of sentiment analysis [3, 4].

Sentiment analysis is another approach that has emerged as a complementary tool in stock trading. Research has shown that investor beliefs about a company's future have a direct effect on buying and selling behavior, so something like media negativity correlates with downward trends in the market [3]. However, sentiment analysis alone has been shown to underperform historical stock data. Without some sort of price context to correlate the sentiment scores too, it cannot accurately generate tradable signals. This study investigates whether combining the sentiment scores from FinBERT with technical price features in an LSTM model will improve risk adjusted trading performance, as well as whether or not the effect differs between high volatility stocks, news driven stocks, and stable macro driven assets.

1.1 Hypothesis

Integrating FinBERT sentiment analysis into an LSTM stock trading model will improve the risk adjusted performance (Sharpe Ratio) compared to a purely technical indicator baseline model, with differing amounts of improvement between high volatility, news sensitive, and stable macro driven stocks.

1.2 Research Questions

- Will the LSTM model with FinBERT sentiment scores have a higher walk forward Sharpe ratio compared to the baseline across all seven stocks?
- Will the type of stock (high volatility, news driven, and macro driven) play a role in how the FinBERT scores positively or negatively affect the overall performance and to what degree?

2 Background Research

Stock prediction has advanced through multiple different eras. Early on, statistical approaches assumed linear relationships in price data, which caused them to struggle with the non-stationary aspect of financial markets. The advancement of machine learning brought models capable of capturing the nonlinear patterns, which were then advanced further by deep learning networks like LSTM [4]. What differentiated these networks from previous methods was that they are able to retain temporal dependencies across long periods of time [7]. Implementation of sentiment scores has begun to become more popular as a sort of addition to the technical data approaches already in place.

Mittal and Goel [6] showed that public mood and stock market movements are correlated through the use of Twitter sentiment. They also showed that sentimental data alone underperforms when

used without historical price data. This finding of theirs made a huge impact on the design of this study, combining technical features with sentiment scores rather than testing on them separately. Wu et al. [7] introduced the S_I_LSTM framework, which combined multiple data sources, including sentiment scores with LSTM networks, and showed that qualitative sentiment signals combined with quantitative price data produce stronger predictions than either of them on their own.

Gu et al. [4] focused their study on FinBERT-LSTM architectures, which found that FinBERT's finance pretraining made it more effective than general-purpose models like VADER for financial news. This is why this study used FinBERT over other models like VADER. Das et al. [3] showed that sentiment effects tend to be asset specific and strongest for individual stocks with high new coverage by providing a survey of sentiment augmented stock prediction methods. This is similar to the findings shown in this study. Gülmez [5] gave insight into how critical it is that you be careful with hyperparameter tuning for LSTM performance, which is why Optuna was used to do an automated search.

Overall walk forward validation has been shown to be more realistic as an evaluation approach than other methods like single train/test splits when it comes to financial machine learning [1, 2]. Since financial data isn't stationary and the fact that a model trained on one period may fail in another, walk forward validation addresses this issue by training on expanding historical windows and then testing on consecutive future periods to better simulate a real-world trading environment.

3 Methodology

3.1 Data Collection

Five-minute OHLCV (open, high, low, close, volume) bar data was collected for seven different stocks (SPY, TSLA, NVDA, MSTR, AAPL, MSFT, and JNJ) using Alpaca's paid market data API which spanned from January 2018 through March 2026. Data was originally stored in CSV files, then eventually switched to Parquet files. SPY was used both as a ticker and as a deciding factor in the trading of the other stocks. Since SPY is an ETF that tracks the S&P500, I decided that if the price is above (bullish) the past 50-day exponential moving average (EMA), then buy/sell signals can be done for other stocks, but if it is below (bearish), then all buy/sell signals are put to a halt and set to hold. News headlines for each ticker were fetched using Alpaca's paid news API.

3.2 Feature Engineering

Both models share 24 technical features gathered from OHLCV data, volatility, trend direction, volume behavior, trend direction, temporal context, and price momentum. The sentiment analysis enhanced model adds one additional feature, which is the FinBERT sentiment score from five-minute bars. Figure 1 below shows feature importance averaged across all 7 tickers. The higher the score, the more important the feature is.

3.3 Model Architecture

Both models use the same neural network structure. The network is made up of one to three LSTM layers stacked on top of one another so that each layer passes what it has learned to the next layer. After the LSTM layers, the data then goes through a set of processing steps before reaching the final output layer, which then produces one of the three trading decisions (BUY, SELL, HOLD). One of the most important steps used was batch normalization. What this does is basically keep the numbers inside the network in a consistent range during training so that abnormalities like the large swings of 2020–2022 would not have such a drastic effect on the training. This helps the models continue to learn stable patterns without these major discrepancies. Hyperparameters were then tuned individually per ticker using an automated hyperparameter search.

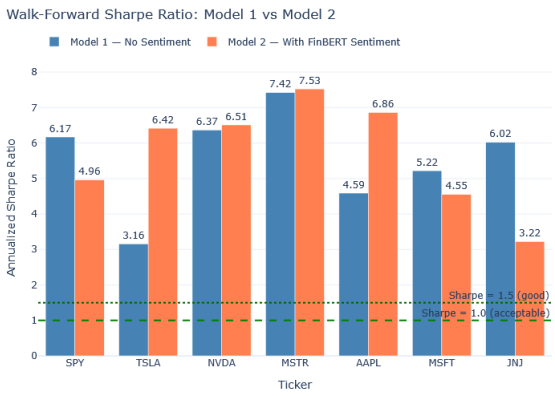

Figure 1: Feature Importance across all seven tickers. Model 1 (Blue – No Sentiment), Model 2 (Orange – Sentiment Included). The higher the value, the more important the feature is.

3.4 Walk Forward Validation

Across five folds, a walk forward scheme was used because of the effects shown in prior research [1, 2]. Each fold was trained on all of the data up to a point where it was cutoff and then evaluated on the next two-week window, which expanded each fold. In use cases like this, walk forward is preferred over k-fold since k-fold defies temporal ordering which could cause training on future data. Early stopping was implemented with a patience of five epochs to watch the Sharpe ratio. A final version of the model was then trained on the full 85% of the data supplied.

3.5 Backtesting

A custom backtest evaluated the final version of the models on the remaining 15% of data. Each ticker's stop losses were set by a volatility percentage, which was 2% for SPY, MSFT, AAPL, JNJ, then 4% for NVDA, TSLA, and lastly 8% for MSTR. SPY was then used as a sort of lock that was only set when the price was below its 50-day EMA. Each model simulation was given $10,000 to start with and the models traded their signals by entering long on BUY, exiting on SELL or stop loss, holding through HOLD, and no sort of short positions.

4 Results and Analysis

The results are shown in two different metrics. The first being walk forward Sharpe ratio, which measures the out of sample risk-adjusted performance across the full 2018–2026 dataset. Then the other being the backtest return on the held out 15% of data, which spanned from early 2025 through March of 2026. These two combined give us a complete picture of the results. Walk forward Sharpe shows consistency across market regimes, whereas the backtest shows performance on completely unseen recent data.

4.1 Implementation Details

All of the models were created in Jupyter Notebook using PyTorch for the LSTM network, pandas and NumPy for data and feature engineering, and Optuna for the hyperparameter tuning. FinBERT sentiment scores were then generated using a pretrained model. Hyperparameters tuned, which included the batch size, number of LSTM layers, hidden size, dropout rate, and sequence length, each of them optimized for walk-forward Sharpe ratio. Early stopping was also included with a patience of five epochs.

4.2 Walk Forward Sharpe Ratio

Table 1 summarizes walk forward Sharpe ratios and returns for both models across all seven tickers. All of the Sharpe values are annualized, which means they show performance over a full year. A Sharpe ratio that is above 1.5 is generally considered good or strong since it indicates the strategy generates good returns relative to the risk it is taking on.

Model B (sentiment enhanced) outperformed Model A on four of seven tickers, those being TSLA (+3.26 Sharpe), NVDA (+0.14), MSTR (+0.11), and AAPL (+2.27). Model B underperformed on SPY (-1.21), MSFT (-0.67), and JNJ (-2.80). The average Sharpe was 5.563 for Model A and 5.720 for Model B. The pattern shown from this is that sentiment gains are heaviest on individual stocks with high media coverage, like TSLA or AAPL, which is to be expected since they have more to score on. Then the smallest or negative scores are either macro-driven, such as JNJ or ETFs like SPY. Figure 2 below shows these results visually.

4.3 Backtest Results

Table 2 shows the backtest results on the 15% of data left out of the testing period. This period of time included quite a volatile span across technology and stocks related to cryptocurrency during early 2025.

Four of the seven tickers showed positive returns that beat out the buy and hold. What buy & hold is showing is if the model had bought the stock and held on to it for the entire period. NVDA stood out the most at +53.9% for both models versus the buy & hold at +24.3%. AAPL's sentiment model then showed +11.3% versus the buy & hold's +6.9%, which is consistent with the walk forward findings. SPY also beat the buy & hold at +10.6% versus +8.7%. JNJ returned +48.8% but underperformed its +67.3% buy & hold, which suggests that the model was too slow with its entry timing. TSLA and MSFT were both negative, which reflects the volatile span with the difficult market conditions mentioned prior. Figure 3 below is a visual representation of these results.

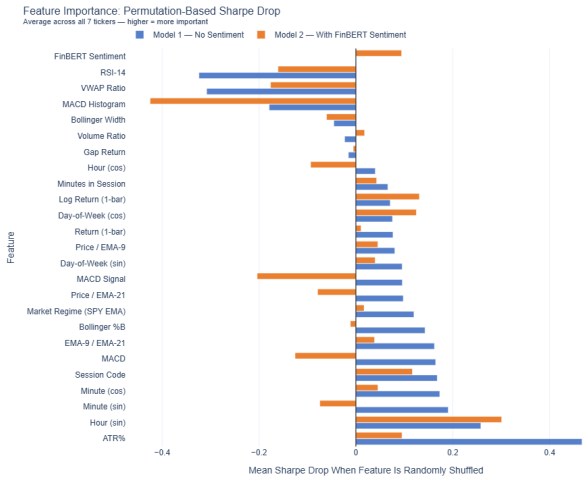

Figure 2: Walk Forward Sharpe Ratio – Model 1(A) vs Model 2(B) across all seven of the tickers. Dotted lines show Sharpe = 1.0 (acceptable) and 1.5 (good) thresholds.

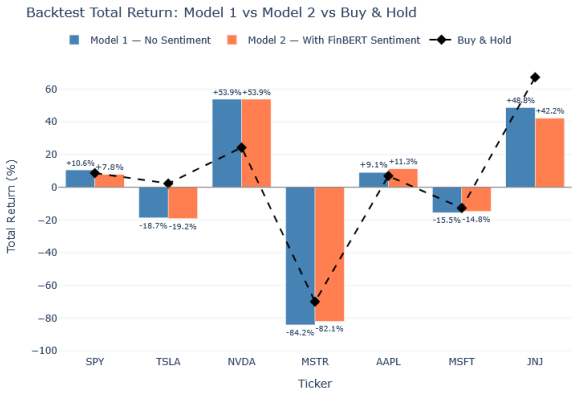

Figure 3: Backtest Total Return – Model 1(A) vs Model 2(B) vs Buy & Hold across the 15% held out test period.

4.4 MSTR and TSLA Analysis

MSTR's -84.2% return came from the models, 21 stop-loss triggers at -8% with a 0%-win rate. This was not necessarily a model failure, but instead just a result of MSTR's drastic downward trend that lasted the entire testing period, and we can see this by the buy % hold being a 70% loss. The model continued to find temporary recoveries as entry signals only for the downward trend to continue on and trigger the stop that is put in place. Then TSLA's -18.7% return despite it's high walk forward Sharpe of 3.156 shows that it was a tough period for trading on this stock. In 2025 and still going into 2026, TSLA faced a large political and brand backlash because the model was not trained on, so it struggled. This is where something like consistent day-to-day or week-to-week training plays a large role in real time trading with models such as these.

Table 1: Walk forward Sharpe ratios and returns. Returns are combined averages across all five folds.

Ticker	M1 Sharpe	M2 Sharpe	M1 Return	M2 Return	Winner
SPY	6.170	4.959	+13.1%	+11.4%	Model 1
TSLA	3.156	6.417	+0.9%	+19.2%	Model 2
NVDA	6.366	6.510	+23.3%	+24.6%	Model 2
MSTR	7.420	7.525	+48.9%	+48.1%	Model 2
AAPL	4.588	6.859	+13.9%	+17.2%	Model 2
MSFT	5.219	4.549	+14.4%	+11.9%	Model 1
JNJ	6.019	3.223	+20.6%	+12.2%	Model 1
Avg	5.563	5.720	+19.5%	+19.2%	Model 2

Table 2: Custom backtest results on held out test period (early 2025–March 2026).

Ticker	M1 Return	M2 Return	B&H Return	M1 Trades	M2 Trades
SPY	+10.6%	+7.8%	+8.7%	3	3
TSLA	-18.7%	-19.2%	+2.3%	4	9
NVDA	+53.9%	+53.9%	+24.3%	1	1
MSTR	-84.2%	-82.1%	-70.0%	21	20
AAPL	+9.1%	+11.3%	+6.9%	7	6
MSFT	-15.5%	-14.8%	-12.7%	7	6
JNJ	+48.8%	+42.2%	+67.3%	4	6

4.5 Sentiment Impact

Figure 4 shows the Sharpe delta (Model 2(B) – Model 1(A)) per ticker. This value reinforces the previous data that showed sentiment analysis helps when company news drives price actions (TSLA +3.26, AAPL +2.27, NVDA +0.14, MSTR +0.11) and hurts when the stock responds mainly to macroeconomic variables (JNJ -2.80, SPY -1.21, MSFT -0.67). This is not only consistent with our other data but also with similar studies that showed sentiment signals are most valuable when a single company's news flow plays a large role in short-term price actions.

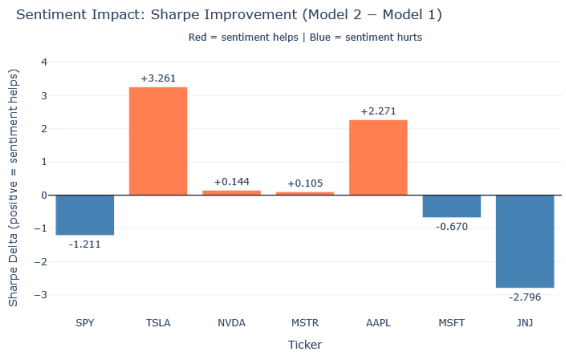

Figure 4: Sentiment Impact – Sharpe Delta (Model 2(B) – Model 1(A)) per ticker. Orange = sentiment helped & blue = sentiment hurts.

5 Conclusions

This study investigated whether FinBERT news sentiment improves LSTM trading model performance compared to a baseline using only technical indicators. The results show that there is a clear pattern where sentiment scores play a part in improving the Sharpe ratio for high volatility, news sensitive stocks (TSLA +3.26, AAPL +2.27) and lowering it for macro-driven assets (JNJ -2.80, SPY -1.21). Model B outperformed Model A on four of the seven tickers with a higher average walk forward Sharpe value (5.720 vs 5.563). FinBERT's finance focused pretraining was shown to be beneficial by the consistent results across the stocks tested on. The five-fold walk forward scheme used was needed to prevent any sort of overfitting that might come from using only a single train/test split on non-stationary financial data.

The next step in this study would be to implement the two models into a framework that uses Alpaca's paid paper trading API. Then running both models simultaneously on separate accounts but identical conditions to provide a direct look into a real-world trading environment that lacks any sort of simulations or anything that can be seen as a bias or advantage. Once that was tested enough, I would then only apply the FinBERT scores to stocks that it has shown to add value and leave it out for those where it didn't or harmed the return. One way the sentiment analysis could be strengthened, which was left out of this study because of the requirements of paid subscriptions, is to include social media (reddit, twitter), SEC filings, and earnings call transcripts that are publicly available. One last step that could be done is testing on different environments, like day-to-day and week-to-week, to see if sentiment scores benefit more in a faster paced setting or more so something like portfolio management over long periods of time.

References

[1] Hamid Arian et al. 2024. Backtest Overfitting in the Machine Learning Era: A Comparison of out-of-Sample Testing Methods in a Synthetic Controlled Environment. *Knowledge-Based Systems* (2024). https://www.sciencedirect.com/science/article/abs/pii/S0950705124011110

[2] Bartosz Bieganowski and Robert Ślepaczuk. 2025. Supervised Autoencoder MLP for Financial Time Series Forecasting. *Journal of Big Data* (2025). doi:10.1186/s40537-025-01267-7

[3] N. Das, B. Sadhukhan, R. Chatterjee, and S. Chakrabarti. 2024. Integrating sentiment analysis with graph neural networks for enhanced stock prediction: A comprehensive survey. *Decision Analytics Journal* 10 (2024), 100417. doi:10.1016/j.dajour.2024.100417

[4] W. Gu, Y. Zhong, S. Li, C. Wei, L. Dong, Z. Wang, and C. Yan. 2024. Predicting Stock Prices with FinBERT-LSTM: Integrating News Sentiment Analysis. 67–72. doi:10.1145/3694860.3694870

[5] B. Gülmez. 2023. Stock price prediction with optimized deep LSTM network with Artificial Rabbits Optimization Algorithm. *Expert Systems with Applications* 227, 120346 (2023), 120346. doi:10.1016/j.eswa.2023.120346

[6] A. Mittal and A. Goel. 2012. *Stock Prediction Using Twitter Sentiment Analysis.* Technical Report. Stanford University. https://cs229.stanford.edu/proj2011/GoelMittalStockMarketPredictionUsingTwitterSentimentAnalysis.pdf

[7] S. Wu, Y. Liu, Z. Zou, and T.-H. Weng. 2021. S_I_LSTM: stock price prediction based on multiple data sources and sentiment analysis. *Connection Science* (2021), 1–19. doi:10.1080/09540091.2021.1940101

Spatiotemporal Detection of Mitotic Events in DNA Fluorescence Time-Lapse Microscopy Using Event-Centric Heatmap Supervision

Andy Yan
andy.yan@go.winona.edu
Department of Computer Science, Winona State University
Winona, Minnesota, USA

Abstract

Accurate detection of mitotic events in time-lapse microscopy is important for quantitative analysis of cell proliferation, but remains challenging in DNA fluorescence imaging because mitotic events are sparse, short-lived, and visually heterogeneous. In this study, mitosis detection is formulated as a spatiotemporal event localization task, in which each event is represented by its spatial center and supervised using Gaussian heatmaps rather than dense masks. Short-term temporal context is incorporated by stacking consecutive frames as input to a lightweight U-Net-based network. The proposed framework combines patch-based training with full-image inference. During training, event-centered patches are sampled to improve learning efficiency under limited annotation. During inference, full-resolution images are processed using an overlapping tile-based strategy, and candidate mitotic locations are recovered from the predicted heatmaps through local maxima detection followed by non-maximum suppression. Experiments were conducted to evaluate the effects of Gaussian scale, temporal window construction, and inference calibration. On the validation split, the proposed method achieved a precision of 0.579, a recall of 0.657, and an F1 score of 0.616, with a mean localization error of 1.87 pixels. These results suggest that center-based heatmap supervision can provide useful localization cues under sparse annotation, although false-positive suppression remains a major challenge in full-image inference.

Keywords

Mitosis detection, spatiotemporal event localization, fluorescence microscopy, Gaussian heatmap supervision, U-Net.

1 Introduction

Mitosis is a central process in cell division and is closely related to abnormal proliferation in cancer. Accurate detection of mitotic events in microscopy data can support the quantitative analysis of cell behavior, proliferation dynamics, and disease-related cellular abnormalities [10, 11, 14, 18]. However, in DNA fluorescence time-lapse microscopy, mitotic events are difficult to identify automatically because they are sparse, short-lived, and visually heterogeneous across frames [2, 10, 17, 18]. As a result, manual annotation is labor-intensive and difficult to scale, particularly for long image sequences and datasets with limited labels [7, 10, 18].

Recent deep learning methods have improved mitosis detection and related microscopy analysis tasks [3, 7, 11, 13]. Many existing approaches formulate the problem as dense segmentation or general object detection. Although these formulations have been successful in related biomedical imaging applications [6, 12, 16, 20], they often depend on dense supervision, such as pixel-level masks or bounding boxes, which is costly to obtain for sparse cellular events [3, 7, 11]. In addition, methods that process frames independently do not explicitly capture the short-term temporal evolution of mitosis, even though the appearance of dividing cells changes over time [15, 17, 18]. These limitations suggest the need for a simpler and more annotation-efficient formulation for mitotic event detection in time-lapse microscopy.

In this study, mitosis detection is formulated as a spatiotemporal event localization task rather than a dense prediction problem. Each mitotic event is represented by its spatial center and supervised using Gaussian heatmaps, which reduces annotation complexity while preserving localization information [21]. Short-term temporal context is incorporated by stacking consecutive frames as input to a lightweight U-Net-based network [12, 15]. To support deployment on high-resolution microscopy data, the proposed framework combines patch-based training with a full-image inference pipeline based on tiled prediction, local maxima detection, and non-maximum suppression.

The main contributions of this work are threefold. First, an event-centric framework is introduced for mitosis localization in DNA fluorescence time-lapse microscopy using center-based Gaussian heatmap supervision and stacked-frame input. Second, an end-to-end workflow is established that connects patch-based training with deployment-time full-image inference. Third, the framework is evaluated through both training-stage ablation studies and inference-stage validation experiments. The results show that the proposed approach can learn useful mitotic localization cues, while also identifying false-positive suppression as a key challenge in full-image inference.

2 Related Work

Deep learning has improved mitosis detection in microscopy images, especially in cases where handcrafted features are not flexible enough to handle the large visual variation of dividing cells. In time-lapse microscopy, the problem is also inherently temporal: mitotic events are sparse, but their appearance changes over neighboring frames rather than within a single image alone [2, 10, 11, 17–19]. For that reason, recent studies have looked beyond static image analysis and explored models for event detection, stage recognition, and temporal refinement [2, 7, 10, 11].

Convolutional neural networks have played a central role in this progress and have become a standard tool in biomedical image analysis [4, 6, 8, 12]. U-Net and related encoder-decoder architectures

are particularly relevant here because they combine contextual feature extraction with relatively precise localization through skip connections [6, 12]. These models have been used widely in medical and microscopy imaging, mostly in segmentation-style settings, and practical tools such as Cellpose and ZeroCostDL4Mic further illustrate the broad adoption of deep learning in microscopy workflows [16, 20].

Temporal modeling has usually been introduced through recurrent designs such as CNN-LSTM or ConvLSTM [15, 17]. Such models are well suited to mitosis detection because a dividing cell can often be recognized more reliably when adjacent frames are considered together rather than in isolation [17, 18]. At the same time, recurrent models add training and implementation complexity, which can become a disadvantage when the dataset is relatively small or the annotation is limited [15, 17].

However, many existing formulations may be unnecessarily complex for sparse mitotic localization. Segmentation methods often assume pixel-level masks, and general detection frameworks usually depend on bounding boxes [3, 7, 11, 13]. For small cellular events, neither form of supervision is ideal. More recent center-based methods suggest a simpler alternative: represent the target by its center and learn a spatial confidence map around that point [21]. That idea fits the present task well. In this study, mitosis is treated as an event-centered localization problem, with Gaussian heatmaps used for supervision and stacked frames used to provide short-range temporal context. This keeps the formulation simpler than recurrent sequence models while still preserving the spatial and temporal information needed for localization.

3 Methodology

Fig. 1 summarizes the overall framework used in this study. The pipeline consists of two stages: patch-based model training and full-image inference. During training, short windows of consecutive DNA fluorescence frames are stacked and fed to a U-Net-based network, while annotated mitotic centers are converted into Gaussian heatmaps for supervision. This training setup keeps the problem local and computationally manageable, which is useful when annotations are limited and mitotic events occupy only a small portion of the full image. At inference time, the trained model is applied to full-resolution microscopy frames through overlapping tiled prediction. The tile-level heatmaps are merged into a full-image response map, and candidate mitotic locations are then extracted by local maxima detection followed by non-maximum suppression.

After the overall workflow is introduced in Fig. 1, it is useful to examine the appearance of the raw microscopy data itself. Fig. 2 provides an example of the raw microscopy data and highlights the sparsity, small size, and visual variability of mitotic events in the full-image setting.

3.1 Problem Formulation

The task in this study is to localize mitotic events in DNA fluorescence time-lapse microscopy by predicting their spatial centers in individual frames. Instead of treating mitosis detection as a dense segmentation problem or a general object detection problem, the method represents each event as a sparse spatiotemporal point. This representation is better suited for the present task because mitotic targets are small, visually variable, and costly to annotate with full masks or bounding boxes [7, 11].

Let a time-lapse sequence be written as a set of grayscale image frames,

$$\{I_t\}_{t=1}^{F}, \tag{1}$$

where F is the number of frames and each $I_t \in \mathbb{R}^{H\times W}$ denotes a two-dimensional image of height H and width W. Let

$$\mathcal{E} = \{(x_i, y_i, t_i)\}_{i=1}^{N} \tag{2}$$

denote the set of annotated mitotic events, where (x_i, y_i) is the spatial center of the i-th event and t_i is the frame in which that event is labeled.

Given a short temporal window of consecutive frames, the model predicts a spatial confidence map for an anchor frame, with higher responses indicating locations that are more likely to contain mitotic centers. These discrete point annotations are then converted into dense supervision through Gaussian heatmaps, as described in the next subsection.

3.2 Heatmap Representation

Sparse point annotations were converted into Gaussian heatmaps to provide spatial supervision for training. Using a single positive pixel at the annotated center would make the target too sharp and fragile, especially for small localization errors. A Gaussian target gives a smoother signal around the event center and preserves local spatial structure, which is more suitable for mitotic localization than direct point labeling. In this setting, the goal is to identify the most likely center of a mitotic event rather than recover its full extent [7, 11, 13, 21].

The model learns a mapping from a short temporal input window to a spatial confidence map,

$$f_\theta : \{I_t\}_{t=k}^{k+T-1} \to \hat{Y}_k, \tag{3}$$

where $\hat{Y}_k$ is the predicted heatmap for the anchor frame. For each frame, the ground-truth heatmap is constructed from the annotated mitotic centers in that frame as

$$Y_k(x, y) = \max_{i:t_i=k} \exp\left(-\frac{(x-x_i)^2 + (y-y_i)^2}{2\sigma^2}\right), \tag{4}$$

where (x_i, y_i) denotes the center of the i-th annotated mitotic event and σ controls the spatial spread of the target response. Smaller values of σ produce sharper targets, whereas larger values yield smoother and more tolerant responses. In this study, σ was treated as a tunable hyperparameter and examined in the ablation experiments.

Fig. 3 shows an example of the heatmap representation used in this work. By replacing discrete point labels with continuous confidence maps, the model is encouraged to learn not only the event center itself, but also the local spatial pattern around it. These heatmaps serve as the supervision targets for the network described below.

3.3 Network Architecture

The model uses a U-Net-based encoder-decoder architecture to predict mitotic localization heatmaps from short temporal windows. Consecutive grayscale frames are stacked along the channel

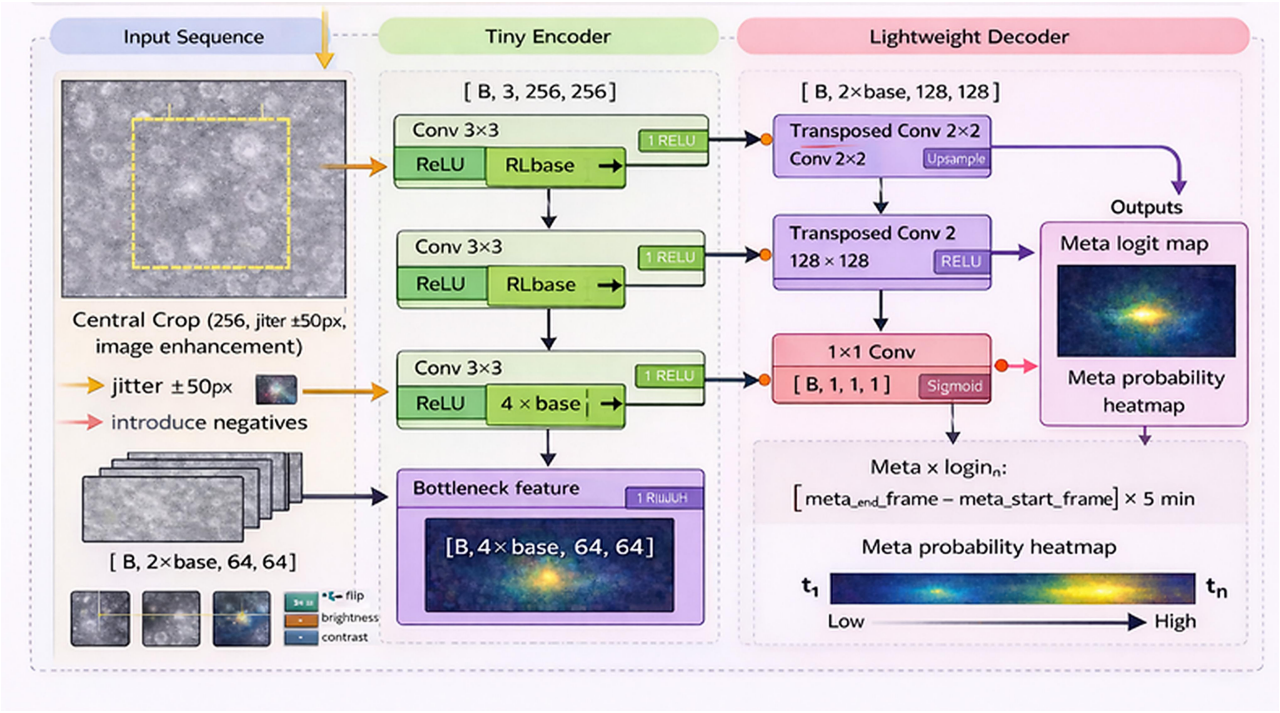

Figure 1: Overview of the proposed spatiotemporal mitosis detection framework, including patch-based training with Gaussian heatmap supervision and full-image inference through tiled prediction, peak extraction, and non-maximum suppression.

dimension and provided as input, so the network can use both spatial appearance and short-range temporal context within a single forward pass. The encoder extracts multiscale features through repeated convolution and down-sampling, while the decoder progressively restores spatial resolution. Skip connections are used to preserve fine localization cues that might otherwise be weakened during down-sampling. The final output is a dense confidence map for the anchor frame, with higher responses indicating locations that are more likely to correspond to mitotic centers.

A U-Net-style architecture was chosen because the task requires both contextual interpretation and precise localization. Mitotic events are small relative to the full microscopy frame, but their local appearance alone is often not sufficient to separate them from visually similar background structures. The encoder-decoder design helps address this by combining broader contextual features with higher-resolution spatial information [6, 12].

Temporal information is introduced through stacked-frame input rather than a recurrent sequence model. This keeps the architecture simpler and easier to train while still allowing the network to use appearance changes across neighboring frames [15, 17]. In this study, the first frame in each temporal window is used as the anchor frame for supervision and prediction.

3.4 Training Objective

The network is trained to produce a spatial response map that matches the Gaussian supervision target for the anchor frame. Let Y denote the ground-truth heatmap and Z the predicted heatmap logits produced by the network. Because dense heatmap prediction is highly imbalanced, positive-region weighting was used to emphasize annotated mitotic regions during training. [9]. The training objective is defined using weighted binary cross-entropy with logits,

$$\mathcal{L}_{\mathrm{BCE}} = -\sum_{x,y}\Big[w(x,y)Y(x,y)\log\sigma(Z(x,y)) + (1-Y(x,y))\log(1-\sigma(Z(x,y)))\Big], \tag{5}$$

where $w(x,y)$ is the positive-region weight and $\sigma(\cdot)$ here denotes the sigmoid activation.

This objective was used because the target is a dense heatmap rather than a binary mask or an object boundary map. The model is therefore encouraged to assign stronger responses near annotated mitotic centers and weaker responses elsewhere.

A practical difficulty is that the target heatmaps are highly sparse. Only a small region around each annotated center carries positive signal, while most pixels belong to the background. Without compensation, the loss would be dominated by easy negative pixels. To reduce this imbalance, positive regions were weighted more heavily than negatives during training. In practice, this made optimization more stable and helped the model learn stronger responses around true mitotic centers while suppressing irrelevant background activations.

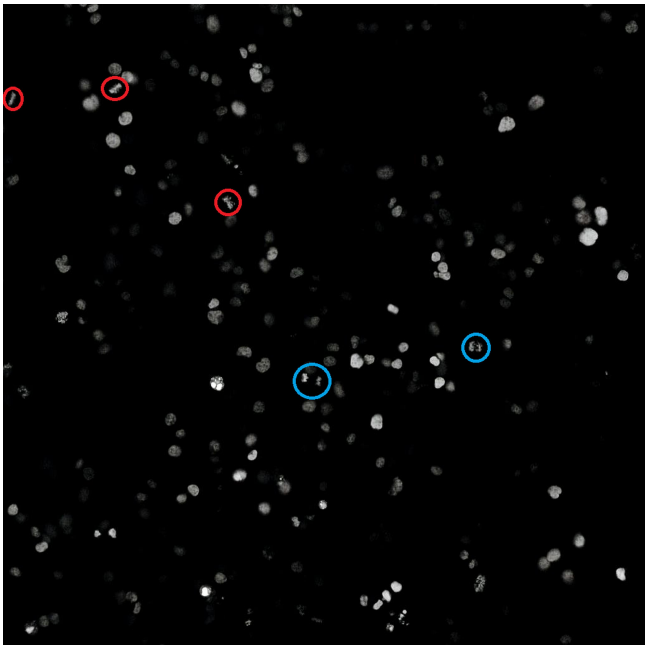

Figure 2: Example DNA fluorescence microscopy frame illustrating the sparse and visually heterogeneous appearance of mitotic events in full-resolution images. Red circles indicate metaphase examples, and blue circles indicate anaphase examples.

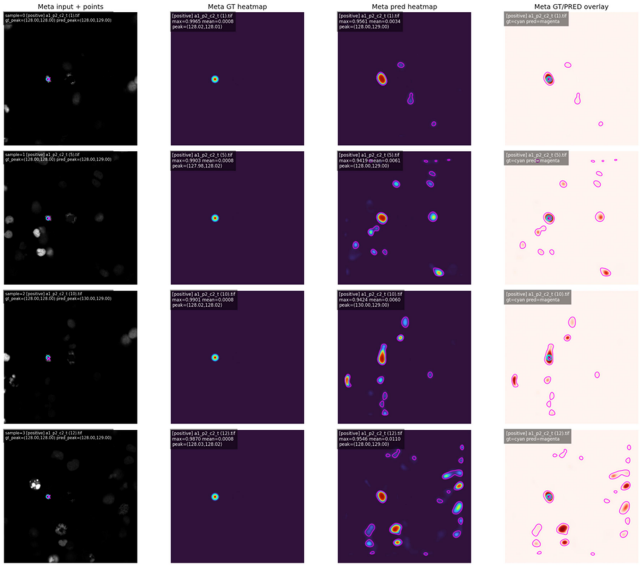

Figure 3: Examples of heatmap-based supervision and prediction for mitotic event localization. From left to right: input microscopy patch with annotated center, ground-truth Gaussian heatmap, predicted heatmap, and ground-truth/prediction overlay.

3.5 Training Strategy

Training was carried out on image patches rather than full-resolution frames. Each sample consisted of a short temporal window of consecutive grayscale frames cropped from the same spatial neighborhood, with the supervision target defined on the anchor frame. This patch-based setup reduced memory cost and kept the model focused on local mitotic patterns, which occupy only a small portion of the full microscopy image.

Different temporal sampling strategies were explored during model development. One used a fixed first-T formulation, in which the earliest frames associated with an event were used to construct a training sample. The other used a sliding-window formulation, which generated multiple temporally overlapping samples from the same event sequence. In the final configuration, the sliding-window strategy was adopted because it provided broader coverage of short-term mitotic dynamics during training.

To reduce positional bias, random spatial jitter was introduced during patch extraction so that the annotated event did not always appear at exactly the same image location. Standard augmentations, including flips and intensity variation, were also applied to increase appearance diversity.

The training set used a mixed sampling strategy with positive patches, randomly sampled negative patches, and hard negative examples. In total, the training data included 798 positive samples, 1596 random negatives, and 559 hard negatives, for 2953 samples overall. The hard negatives were included to expose the model to visually confusing non-mitotic structures that could otherwise lead to false positives during full-image inference. The validation patch set contained 313 samples, all centered on annotated positive events.

3.6 Inference and Post-Processing

Because the model was trained on patches, inference on full-resolution microscopy frames was performed with an overlapping tile-based strategy. Each image was divided into tiles with 96-pixel overlap, and the trained network was applied to each tile separately. The resulting tile-level heatmaps were then merged into a full-image response map. Pyramid blending was used during this step to reduce boundary artifacts and produce smoother transitions between overlapping regions.

Candidate mitotic locations were extracted from the merged heatmap by local-maximum-based peak detection. Responses with scores below 0.28 were discarded. Local maxima were computed with a kernel size of 50 pixels so that weak nearby responses would not generate multiple closely spaced peaks. After peak extraction, non-maximum suppression was applied with a radius of 30 pixels to remove redundant neighboring detections. The remaining peaks were taken as the final mitotic coordinates for that frame.

This post-processing pipeline converts dense heatmap predictions into a set of discrete event locations. In practice, full-image performance depended not only on the quality of the predicted heatmap, but also on how these responses were filtered and consolidated during inference.

Table 1: Main ablation variables and inference calibration settings used to evaluate the proposed mitosis detection framework.

Component	Variants / Values Tested	Purpose
Gaussian scale (σ)	3, 4, 5, 6	Evaluate heatmap spread sensitivity
Temporal window size (T)	2, 3, 4, 5	Assess short-term temporal context
Temporal sampling strategy	first-T, sliding	Compare window construction methods
Training sample composition	798 positive, 1596 random negative, 559 hard negative samples	Improve robustness and reduce false positives during inference
Patch size	256×256	Enable efficient local training on high-resolution microscopy data
Inference mode	Overlapping tile-based prediction	Support deployment on full-resolution frames
Tile size / overlap	256×256 / 96 pixels	Balance spatial coverage and boundary stability during inference
Score threshold	0.15, 0.18, 0.23, 0.25, 0.28, 0.30, 0.35	Calibrate inference peak selection
Fixed inference parameters	local max kernel = 50, NMS radius = 30	Stabilize post-processing

3.7 Experimental Setup and Ablation Design

The framework was evaluated through both training-stage ablation studies and inference-stage validation experiments. The main variables considered in this study are summarized in Table 1. Some of these settings affect how the model learns from patch-based supervision, whereas others determine how full-image heatmap responses are converted into final mitotic coordinates at inference time.

During model development, the main training-side variables were the Gaussian scale used to generate supervision targets, the temporal window size, and the way temporal windows were constructed. Both a fixed first-T strategy and a sliding-window strategy were examined. The goal was not only to compare different temporal input lengths, but also to test whether the way short-term temporal context is sampled affects downstream localization performance.

The composition of the training data was also treated as an important part of experimental design. The final training set included positive patches, randomly sampled negative patches, and hard negative examples. This was done to make the model less sensitive to visually confusing non-mitotic structures, which became a practical issue during full-image inference.

Inference-stage experiments were used to calibrate the final deployment pipeline on full-resolution microscopy frames. After the model had been trained, additional validation runs were carried out to determine a suitable score threshold for peak selection and to assess the effect of the maximum number of retained peaks per frame. By contrast, the local-maximum kernel size and the non-maximum suppression radius were kept fixed once a stable operating range had been identified.

Unless otherwise noted, the framework used patch-based training with Gaussian heatmap supervision, stacked temporal input, and full-image inference through overlapping tiled prediction followed by local maxima detection and non-maximum suppression. The evaluation metrics used to assess these experiments are described in the next subsection.

Table 2: Event-level detection and localization performance of the proposed framework on the validation split under the final full-image inference setting.

Metric	Value
Precision	0.579
Recall	0.657
F1 Score	0.616
Mean Localization Error (px)	1.873
Hit@5	0.981
Hit@10	1.000
Hit@15	1.000
Number of Matched Events	686
Number of Processed Frames	843

3.8 Evaluation Metrics

Performance was evaluated at the event level by matching predicted mitotic coordinates to ground-truth annotations within a spatial tolerance of 30 pixels. A prediction was counted as a true positive if it could be matched to an annotated mitotic event within this radius. Predictions without a match were counted as false positives, and annotated events that were not detected were counted as false negatives.

Based on these matches, precision, recall, and F1 score were used to summarize detection performance. Precision reflects how many predicted events corresponded to true mitotic events, whereas recall measures how many annotated events were successfully recovered. The F1 score was used to summarize the balance between these two quantities.

Detection performance alone does not fully describe localization quality, so additional spatial accuracy metrics were also reported. For matched detections, the Euclidean distance between the predicted center and the corresponding ground-truth center was measured, and the mean of these distances was reported as the mean localization error. Hit@5, Hit@10, and Hit@15 were also included to indicate the proportion of matched detections whose localization error fell within 5, 10, and 15 pixels, respectively. Together, these metrics provide a more complete view of both detection performance and spatial localization accuracy.

4 Results and Analysis

4.1 Full-Image Inference Performance

The final framework was evaluated on the validation split using overlapping tiled prediction and post-processing. Since the main purpose of this study was to test whether a patch-trained localization model could remain effective in full-resolution microscopy scenes, the event-level detection and localization metrics under the selected inference setting are the primary results. The overall validation performance is reported in Table 2.

As shown in Table 2, the final configuration achieved a precision of 0.579, a recall of 0.657, and an F1 score of 0.616 on the validation split. These values indicate that the model was able to recover a substantial portion of annotated mitotic events under full-image inference, although false positives remained a noticeable source of error. The localization metrics were stronger than the detection metrics: the mean localization error was 1.87 pixels, and both Hit@10 and Hit@15 reached 1.000. This indicates that, when a true mitotic event was successfully detected, its predicted center was usually close to the annotation.

Taken together, these results suggest that the proposed framework learned meaningful localization cues, but that deployment on full-resolution images remained more difficult than patch-level learning alone. The main limitation was not spatial precision after a correct detection had been made, but the ability to suppress background responses that resembled true mitotic events under realistic full-image conditions.

4.2 Training-Stage Ablation Results

Table 2 reports the final validation performance under the selected full-image inference setting. Before that operating point was chosen, a series of training-stage ablation experiments was carried out to determine a reasonable model configuration. These experiments focused on design choices that most directly affect heatmap learning and short-range temporal modeling, especially the Gaussian scale used for supervision and the way temporal windows were constructed during training. The results are presented in the following subsections.

4.2.1 Gaussian Scale Study. The Gaussian scale controls how sharply each annotated mitotic center is represented in the supervision target. This directly affects the balance between localization specificity and tolerance to small spatial deviations during training. To examine its effect, several values were tested under the same baseline configuration, and the results are summarized in Table 3.

A clear trade-off can be seen across the tested settings. Smaller values produced sharper supervision targets and tended to improve precision, but they also made the model more conservative and reduced recall. Larger values had the opposite effect: recall increased because the supervision became spatially smoother, but this also introduced more false responses and lowered precision. In other words, overly sharp targets made the detector less tolerant, whereas overly broad targets weakened spatial selectivity.

Among the tested settings, $\sigma = 5$ gave the best overall balance and achieved the highest F1 score. Although $\sigma = 6$ produced the highest recall, that gain was offset by reduced precision. By contrast, $\sigma = 4$ achieved the highest precision but missed more annotated events. Localization quality remained strong across all three settings, with nearly identical Hit@K values, although the mean localization error increased slightly at the largest σ. Based on this comparison, a moderate Gaussian scale was retained in the final configuration.

4.2.2 Temporal Window and Sampling Strategy. To examine how short-term temporal information should be introduced during training, two temporal sampling strategies were compared. The first used a fixed first-T formulation, in which each sample was constructed from the earliest frames associated with an event. The second used a sliding-window formulation, which generated multiple temporally overlapping samples from the same event sequence. Validation experiments were then carried out with different window lengths to assess how these design choices affected downstream detection performance.

Table 3: Comparison of Gaussian scale (σ) settings under the baseline configuration. Event-level detection and localization metrics are reported on the validation split.

σ	Prec.	Rec.	F1	MLE (px)	H@5	H@10 / H@15
4	0.662	0.555	0.604	1.75	0.981	1.000 / 1.000
5	0.579	0.657	**0.616**	1.87	0.981	1.000 / 1.000
6	0.510	0.720	0.597	2.20	0.981	1.000 / 1.000

Note: Prec. = precision; Rec. = recall; MLE = mean localization error; H@K = Hit@K.

Table 4: Comparison of temporal input construction strategies on validation performance.

Strategy	Window (T)	Stride	Prec.	Rec.	F1
first-T	3	–	0.507	0.552	0.529
first-T	4	–	0.531	0.524	0.527
sliding	3	3	0.579	0.657	0.616
sliding	4	4	0.496	0.589	0.538

Note: Prec. = precision; Rec. = recall.

The results in Table 4 show that temporal input construction mattered beyond the choice of window length alone. Across the tested settings, the sliding-window formulation produced consistently stronger recall than the fixed first-T strategy. This suggests that the model benefited from seeing a broader range of short-term appearance changes during training, rather than being restricted to a single fixed temporal view of each event.

The best overall performance was obtained with the sliding-window strategy at $T = 3$, which achieved the highest F1 score among the compared settings. Increasing the window length to $T = 4$ did not improve performance further, indicating that adding more frames was not necessarily beneficial in this setting. A plausible explanation is that short mitotic dynamics were already captured within a three-frame context, whereas longer windows introduced additional variation without providing equally useful information.

Overall, these results indicate that temporal modeling was helpful, but the way temporal context was sampled during training was at least as important as the number of frames included in the input. Based on this comparison, the sliding-window strategy with $T = 3$ was adopted in the later configuration.

4.3 Inference Calibration

Full-image performance depended not only on what the model learned during training, but also on how the predicted heatmaps were converted into discrete mitotic detections at inference time. Because the final output was produced through peak extraction and suppression steps, several inference-side parameters affected the balance between recovering true mitotic events and filtering

Table 5: Validation performance under different score-threshold settings during full-image inference.

Threshold	Prec.	Rec.	F1	MLE (px)
0.15	0.489	0.731	0.586	2.039
0.18	0.506	0.719	0.594	1.991
0.23	0.537	0.705	0.610	1.941
0.25	0.547	0.690	0.610	1.924
0.28	0.579	0.657	**0.616**	1.873
0.30	0.565	0.654	0.606	1.843
0.35	0.582	0.631	0.605	1.815

Note: Prec. = precision; Rec. = recall; MLE = mean localization error.

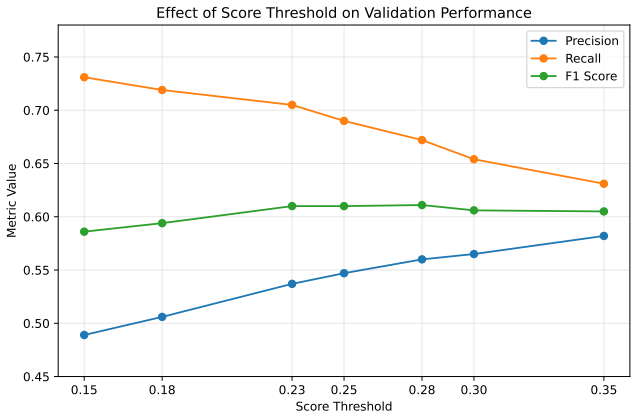

Figure 4: Effect of score threshold on validation performance during full-image inference. Increasing the threshold generally reduces recall, while precision tends to improve with small fluctuations. F1 reaches its best balance in the intermediate threshold range.

spurious responses. The main factors examined here were the score threshold used for peak selection and the maximum number of peaks retained per frame. These experiments were carried out on the validation split using the same trained model so that the effect of post-processing could be assessed separately from changes in model training. The corresponding results are presented in the following subsections.

4.3.1 Score Threshold Study. The score threshold determines which heatmap responses are retained as candidate detections after peak extraction. In practice, it acts as an operating-point control for the final inference pipeline. Lower thresholds preserve more responses and therefore tend to improve recall, but they also allow more weak or spurious peaks to survive. Higher thresholds have the opposite effect: false responses are suppressed more aggressively, but some true mitotic events are also lost.

This trend can be seen clearly in Table 5, and the same pattern is visualized in Fig. 4. At the low end of the tested range, recall was highest, but precision was reduced by the larger number of false positives. As the threshold increased, recall generally declined, while precision tended to improve despite small fluctuations. F1 reached its highest value in the intermediate threshold range rather than at either extreme.

Among the tested settings, a threshold of 0.28 gave the best overall F1 score and was retained in the final configuration. Thresholds below this value kept more candidate events but allowed too many false responses, whereas higher thresholds became increasingly restrictive and started to remove valid detections. Fig. 4 makes this trade-off easier to see by showing that the threshold mainly changes the operating point of the detector rather than the underlying localization behavior of the model.

Overall, the threshold sweep shows that full-image inference performance depended not only on the learned heatmap quality, but also on how the response map was converted into final event coordinates. For that reason, threshold selection was treated as part of the inference design rather than as a minor implementation detail.

4.3.2 Maximum Peak Count Study. In addition to threshold selection, full-image inference was also affected by the maximum number of peaks retained in each frame after post-processing. This parameter limits how many candidate responses are allowed to survive after peak extraction and suppression. A smaller peak budget makes the detector more conservative, whereas a larger one preserves more candidate events at the cost of additional false responses.

Table 6: Validation performance under different maximum-peak settings during full-image inference.

Max Peaks	Prec.	Rec.	F1	MLE (px)
2	0.610	0.617	0.614	1.855
3	0.579	0.657	**0.616**	1.873
4	0.565	0.665	0.611	1.867
5	0.560	0.672	0.611	1.872

Note: Prec. = precision; Rec. = recall; MLE = mean localization error.

The comparison in Table 6 shows the same general trade-off seen in the threshold study, but in a different form. When only a small number of peaks was retained, precision improved because fewer low-confidence or redundant responses remained in the final output. As the peak budget increased, recall became stronger, indicating that more true mitotic events were preserved. At the same time, the additional retained peaks also introduced more false positives, which gradually reduced precision.

The best overall F1 score was obtained at an intermediate setting rather than at either extreme. This suggests that the final performance did not benefit from making the detector as restrictive as possible or as permissive as possible. Instead, a moderate peak budget provided the most effective balance between recovering true mitotic events and suppressing spurious detections in crowded full-image scenes.

Taken together, the results in Table 6 show that post-processing design remained an important part of the full-image inference pipeline. The quality of the predicted heatmap mattered, but so did the rule used to convert that response map into a limited set of final event coordinates.

Additional sensitivity checks were also performed for the NMS radius and local-maximum kernel size. Within the tested ranges (NMS radius: 15–50 pixels; local-maximum kernel size: 15–60 pixels), these parameters produced only minor changes in the overall metrics compared with the effects of score threshold and maximum peak count. Therefore, fixed values of 30 and 50 were used in the final configuration.

4.4 Discussion

The results support the use of an event-centric formulation for mitosis localization in DNA fluorescence time-lapse microscopy. Representing each event by its center and supervising the model with Gaussian heatmaps was sufficient for learning meaningful spatial responses around annotated mitotic locations. In that sense, dense mask supervision may not be necessary for this form of center-based mitotic localization. For sparse mitotic events, center-based supervision offers a simpler and more annotation-efficient alternative.

At the same time, the experiments also make clear that patch-level learning and full-image inference are not equally difficult. Once a true mitotic event was detected, the predicted center was usually very accurate, as reflected by the low localization error and strong Hit@K performance. The larger difficulty was deciding which responses should be kept in the full-image setting. In other words, the main limitation was not precise spatial localization itself but separating true mitotic responses from visually similar background structures under realistic inference conditions.

The ablation and calibration experiments help explain this gap. On the training side, both the Gaussian scale and the temporal input strategy affected downstream performance. In particular, the comparison between fixed first-T windows and sliding temporal windows suggests that the model benefited from seeing a broader range of short-term mitotic dynamics during training. On the inference side, full-image performance depended strongly on thresholding and peak-budget selection, while the NMS radius and local-maximum kernel size had comparatively smaller effects. This pattern suggests that the model had already learned useful local response patterns, but the final detection quality still depended on how those responses were filtered and consolidated during post-processing.

Several limitations also emerged from this study. First, the current circular heatmap target may not be the best match for mitotic DNA structures, which often appear elongated or bar-like rather than approximately isotropic. A more shape-aware target, such as an elliptical heatmap, may better reflect the underlying morphology and improve supervision quality. Second, the annotations are likely incomplete in densely populated regions. As seen in Fig. 2, some crowded areas contain highly similar DNA structures, making exhaustive manual labeling difficult. As a result, some detections currently considered false positives may in fact correspond to real but unlabeled mitotic events, which would lead to an underestimation of true precision.

Another limitation is that the present analysis has focused mainly on localization and has not yet examined anaphase in the same depth as metaphase. Extending the framework to include anaphase more explicitly would provide a more complete view of mitotic progression. More broadly, the long-term goal of this work is not only to localize mitotic events, but also to estimate mitotic duration as a quantitative indicator of abnormal cancer-cell division. Reaching that goal will require more than point localization alone. It will likely depend on more complete event association across frames, explicit phase modeling, and better temporal interpretation of the mitotic process [1, 5].

5 Conclusion

This study presented an event-centric framework for mitosis localization in DNA fluorescence time-lapse microscopy. Instead of treating the task as dense segmentation, the proposed method represented each mitotic event by its spatial center and used Gaussian heatmap supervision together with stacked temporal input to learn localization cues. The framework combined patch-based training with full-image inference, making it possible to train efficiently while still evaluating the model under realistic full-resolution conditions.

The experimental results showed that this formulation was able to learn meaningful mitotic responses and localize matched events with high spatial accuracy. The ablation experiments further indicated that both supervision design and temporal input construction affected downstream performance, while the inference studies showed that post-processing choices remained important in full-image deployment. These findings suggest that event-centric heatmap learning is a practical and annotation-efficient direction for mitosis localization, while also showing that false-positive suppression remains a central challenge in crowded microscopy scenes.

Several directions remain for future work. A more shape-aware supervision target, such as an elliptical heatmap, may better match the morphology of mitotic DNA than the circular targets used here. Improved annotation quality will also be important, especially in densely populated regions where some detections may correspond to real but unlabeled events. In addition, the present study has focused mainly on localization and has not yet examined anaphase as fully as metaphase. Extending the framework to model a broader range of mitotic stages, and ultimately to estimate mitotic duration, will be an important next step.

Overall, this work shows that event-centric heatmap learning is a promising direction for mitosis localization. It also highlights the remaining gap between accurate local response learning and robust full-image deployment.

References

[1] M. Awais et al. 2024. BOrg: A brain organoid-based dataset for mitosis detection. *arXiv preprint arXiv:2406.19556* (2024).

[2] T. Dincer, J. Stegmaier, and A. Jose. 2024. Identification of mitosis stages using artificial neural networks for 3D time-lapse cell sequences. *bioRxiv* (2024).

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2961–2969.

[4] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[5] S. He et al. 2024. D-MAINS: A deep-learning model for label-free detection of mitosis and cell states. *Cells* (2024).

[6] F. Isensee, J. Petersen, A. Klein, D. Zimmerer, P. F. Jaeger, S. Kohl, J. Wasserthal, G. Koehler, T. Norajitra, S. Wirkert, and K. H. Maier-Hein. 2018. nnU-Net: Self-adapting framework for U-Net-based medical image segmentation. *arXiv preprint arXiv:1809.10486* (2018).

[7] M. Jahanifar, A. Shephard, N. Zamanitajeddin, S. Graham, S. E. A. Raza, F. Minhas, and N. Rajpoot. 2024. Mitosis Detection, Fast and Slow: Robust and Efficient Detection of Mitotic Figures. *Medical Image Analysis* 94 (2024), 103132. doi:10.1016/j.media.2024.103132

[8] A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2980–2988.

[10] Y. Mao, L. Han, and Z. Yin. 2019. Cell mitosis event analysis in phase contrast microscopy images using deep learning. *Medical Image Analysis* 57 (2019), 32–43.

[11] C. Piansaddhayanon, S. Santisukwongchote, S. Shuangshoti, Q. Tao, S. Sriswasdi, and E. Chuangsuwanich. 2023. ReCasNet: Improving Consistency within the Two-Stage Mitosis Detection Framework. *Artificial Intelligence in Medicine* 135 (2023), 102462. doi:10.1016/j.artmed.2022.102462

[12] O. Ronneberger, P. Fischer, and T. Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 234–241.

[13] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers. 2018. Cell detection with star-convex polygons. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 265–273.

[14] Z. Shen, M. Simard, D. Brand, et al. 2024. A deep learning framework deploying segment anything to detect pan-cancer mitotic figures. *Communications Biology* 7 (2024), 1674.

[15] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*, Vol. 28. 802–810.

[16] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu. 2021. Cellpose: A generalist algorithm for cellular segmentation. *Nature Methods* 18, 1 (2021), 100–106.

[17] Y.-T. Su, Y. Lu, M. Chen, and A.-A. Liu. 2017. Spatiotemporal joint mitosis detection using CNN–LSTM network in time-lapse phase contrast microscopy images. *IEEE Access* 5 (2017), 18045–18056.

[18] Y.-T. Su, Y. Lu, J. Liu, M. Chen, and A.-A. Liu. 2021. Spatio-temporal mitosis detection in time-lapse phase-contrast microscopy image sequences: A benchmark. *IEEE Transactions on Medical Imaging* 40, 5 (2021), 1319–1328. doi:10.1109/TMI.2021.3052854

[19] J. Turley et al. 2024. Deep learning for rapid analysis of cell divisions in vivo. *eLife* (2024).

[20] L. von Chamier, R. F. Laine, J. Jukkala, C. Spahn, D. Krentzel, E. Nehme, M. Lerche, S. Hernández-Pérez, P. K. Mattila, E. Karinou, S. Holden, A. C. Solak, A. Krull, T.-O. Buchholz, M. L. Jones, L. A. Royer, C. Leterrier, Y. Shechtman, F. Jug, M. Heilemann, G. Jacquemet, and R. Henriques. 2021. Democratising deep learning for microscopy with ZeroCostDL4Mic. *Nature Communications* 12, 1 (2021), 2276. doi:10.1038/s41467-021-22518-0

[21] X. Zhou, D. Wang, and P. Krähenbühl. 2019. Objects as points. *arXiv preprint arXiv:1904.07850* (2019).