Kuwait Journal of Advanced Computer Technology
ISSN: 2984-7907
Vol 4 Issue 1 (2026)

An Analytical Survey of Cyberbullying Detection Using Machine Learning Algorithms

Nikesh Aote[1], Sheron Sheikh[2], Soheb Pathan[3], Sohel Sayyad[4], Akash Bisen[5], Sumit kalbande[6]

[1]Assistant Prof, Computer science and engineering

Priyadarshini college of engineering, Nagpur, India

[2]Computer science and engineering,

Priyadarshini college of engineering, Nagpur, India

[3]Computer science and engineering,

Priyadarshini college of engineering, Nagpur, India

[4]Computer science and engineering,

Priyadarshini college of engineering, Nagpur, India

[5]Computer science and engineering,

Priyadarshini college of engineering, Nagpur, India

[6]Computer science and engineering,

Priyadarshini college of engineering, Nagpur, India

Submitted: 03/12/2025 **Revised**: 15/12/2025 **Published**: 30/12/2025

Abstract

Moderating harmful text directly from dynamic desktop screens requires converting pixels into textual content reliably and classifying that text accurately, all under tight latency and privacy constraints. This survey integrates research across three pillars to guide the design of on-device moderation systems: (i) optical character recognition (OCR) for heterogeneous user interfaces (UIs), including neural (LSTM-based) OCR, scene text detectors, and sequence recognition models; (ii) abusive/toxic language detection methods that range from lexicon rules to supervised transformers and zero-shot classification framed as natural language inference (NLI); and (iii) system-level design strategies, such as high-throughput screen capture, region-of-interest (ROI) scheduling, frame skipping, GPU-aware inference, and efficient overlay compositing. We prioritized peer-reviewed venues and canonical documentation in selecting sources. The review finds that: OCR fidelity is the principal ceiling for downstream moderation; hybrid pipelines combining lexicons with context-aware transformers typically outperform single-signal approaches; and zero-shot models broaden label coverage and cross-lingual generalization but require threshold calibration and bias auditing. Significant gaps remain in handling code-mixed Hindi–English text, stabilizing OCR on stylized UI renderings, and mitigating unintended biases across user groups. The survey concludes with practical engineering guidance, an evaluation blueprint (accuracy, latency, and fairness), and research directions for robust, privacy-preserving on-device moderation [1]–[14].

1. Introduction

On-device moderation of visual interfaces is fundamentally different from filtering server-side logs or API payloads. Desktop applications present text within heterogeneous UIs—browsers, chats, productivity tools—where fonts, backgrounds, languages, and rendering effects (e.g., anti-aliasing, transparency) vary widely. A workable moderation pipeline must therefore: (1) capture screen pixels efficiently; (2) extract text reliably with OCR; and (3) assess toxicity with calibrated precision while preserving real-time responsiveness.

Advances in OCR—from pattern-based recognition to LSTM-based neural engines—have substantially improved line-level accuracy and script coverage [1], [2]. Scene text detectors offer fast, accurate region proposals that improve recognition in cluttered scenes [3]. Sequence recognition models (e.g., CRNN with CTC)

further increase robustness to variable-length words and distortions [4]. In parallel, moderation approaches have evolved from lexicon rules to supervised transformers (context-aware toxicity scoring) and zero-shot NLI-based classification for label flexibility and cross-lingual generalization [5]–[8]. Surveys on hate speech underscore enduring challenges: ambiguity, obfuscation, domain shift, and fairness [9], [10]. For Hindi and Indo-European languages, HASOC-style resources highlight code-mixed complexities that stress both OCR and language models [11].

This survey synthesizes these bodies of work for an engineering audience. It emphasizes practical choices—ROI scheduling, frame skipping, GPU utilization, overlay design—that materially influence end-to-end performance and user experience.

2. Methodology

- Scope and focus: Works directly applicable to on-screen OCR and textual toxicity detection, with emphasis on multilingual and code-mixed contexts, and real-time system design.

- Sources: IEEE Xplore, CVF Open Access (CVPR), ACL Anthology/EMNLP, ACM Digital Library, arXiv, and official framework/model documentation for widely adopted tools.

- Keywords: “Tesseract LSTM OCR,” “EAST scene text detector,” “CRNN recognition CTC,” “toxic-bert,” “DeBERTaV3,” “XNLI,” “zero-shot classification,” “hate speech survey NLP,” “HASOC Hindi dataset,” “Python MSS,” “UpdateLayeredWindow.”

- Inclusion criteria: Technical relevance to OCR on heterogeneous UIs; evidence of robustness across languages; moderation methods deployable on device; and canonical system-level references for screen capture/overlays.

- Exclusion criteria: Non-text modalities (e.g., audio-only); highly domain-specific corpora with narrow applicability; works lacking methodological transparency.

- Analysis approach: Thematic synthesis (OCR → moderation → system), identification of strengths/weaknesses, discussion of trade-offs and deployment concerns, and alignment with real-time constraints.

3. Literature Review

3.1 OCR Engines for Heterogeneous UIs

Neural OCR has matured with the incorporation of recurrent networks, as in modern Tesseract releases, which improved line-level recognition and Unicode coverage compared to legacy engines [1], [2]. For on-screen content, three factors are particularly salient:

- Language hints: Specifying multiple languages (e.g., Hindi + English) reduces script-level confusion and improves tokenization.

- Preprocessing: Lightweight steps (grayscale, de-noising, adaptive thresholding, contrast stretching) stabilize recognition for anti-aliased, low-contrast text common in UIs.

- Region selection: Constraining OCR to likely text regions reduces false positives and computational overhead.

3.2 Scene Text Detection and Localization

EAST (Efficient and Accurate Scene Text) demonstrates a fast and accurate method for localizing text via fully convolutional networks and geometry predictions [3]. The detector-before-OCR paradigm is attractive for screenshots because it:

- Filters background clutter, images, and icons that confound OCR;

- Improves effective OCR accuracy by presenting text-dominant crops;

- Enables confidence-weighted cascades where weak proposals are downweighted or ignored.

3.3 Sequence Recognition Models

CRNN-style recognizers view text as sequences and use CTC decoding to avoid explicit character segmentation [4]. This design improves robustness to distortions, variable-length tokens, and imprecise character boundaries—conditions frequent in desktop UIs with scaling and anti-aliasing. Compact CRNN variants and quantized deployments are increasingly practical for desktop-time constraints.

3.4 Lexicon-Based Toxicity Detection

Lexicon filters remain useful for explicit slurs and controlled vocabularies. Strengths include high precision on known abusive terms, straightforward maintenance of variants/transliterations, and predictable behavior. Weaknesses include lack of context (e.g., quotes, negations, satire) and vulnerability to obfuscation. Practical implementations incorporate cooldowns to prevent repeated triggers for persistent on-screen text and normalization to handle lookalike characters.

3.5 Supervised Transformers for Toxicity

Transformer-based classifiers trained on large annotated corpora (e.g., Jigsaw challenges) provide context-aware assessments that often outperform lexicon rules alone [5]. Deployment considerations include:

- Threshold calibration: Selecting operating points per label to balance false positives and recall for the application's risk profile.

- Domain shift: UI text can differ from social media comments; prompt/label engineering or light adaptation can reduce mismatch.

- Calibration and interpretability: Confidence scores benefit from post-hoc calibration; explanations can build user trust.

3.6 Zero-Shot Classification via NLI

Zero-shot classification reframes detection as an NLI task: each label is a hypothesis, and the model evaluates entailment given the input text [8]. DeBERTaV3 improves efficiency and downstream performance with ELECTRA-style pretraining and gradient-disentangled embedding sharing [6]. Cross-lingual generalization is frequently evaluated with XNLI [7]. Operationally:

- Label engineering matters: Clear, concise label descriptions typically improve results.

- Multi-label configuration: Independent thresholds per label capture overlapping categories (e.g., "insult," "hate," "threat").

- Calibration: Per-label calibration and threshold sweeps stabilize performance across domains and languages.

3.7 Multilingual and Code-Mixed Contexts

HASOC-style tasks and datasets highlight the complexities of Hindi and related Indo-European languages [11]. Key observations:

- Code-mix and transliteration: Mixed scripts (Devanagari/Latin) and creative spellings stress both OCR and classifiers.

- Annotation ambiguity: Cultural nuances and differing guidelines complicate labels' boundaries, affecting reliability.

- Domain mismatch: Social media datasets may not reflect desktop UIs; collecting representative evaluation samples is essential for deployment.

3.8 System-Level Components for Real-Time Use

- High-throughput capture: MSS is designed for fast, cross-platform screen grabbing with minimal overhead [12].

- Efficient overlays: Windows layered windows (UpdateLayeredWindow) support per-pixel alpha composition, enabling blur/mask overlays without forcing repaints of underlying applications [13].

- Practical UI: Toolkits like Kivy/KivyMD can present status, toggles, notifications, and logs with minimal boilerplate and acceptable performance [14].

- Scheduling and throttling: ROI grid sweeps and frame skipping provide predictable compute budgets at stable latencies.

- Hardware utilization: GPU offload can reduce inference time for transformers; otherwise, quantization and batching are effective on CPU.

4. Comparative Analysis and Trade-offs

4.1 OCR Design Choices

- Detector-before-OCR vs. direct OCR: A detector-first pipeline (e.g., EAST then OCR) reduces clutter and boosts OCR accuracy at the cost of an extra model stage. Direct OCR on the whole frame is simpler but pays in accuracy and compute.

- LSTM OCR vs. CRNN recognition: LSTM-based OCR is convenient and widely available; CRNNs can outperform in challenging conditions if trained/tuned, but add model management complexity.

- Preprocessing depth: Aggressive preprocessing can yield gains on anti-aliased or low-contrast UI text but risks removing weak strokes if not tuned.

4.2 Moderation Logic

- Lexicon-only: High precision for explicit terms, low recall for implicit/obfuscated cases.

- Transformer-only: Context-aware with better recall, but can over-flag without careful thresholds and calibration.

- Hybrid: Combining lexicon cues with transformers (and zero-shot augmentation) typically increases reliability, especially when outputs are fused with cooldowns and multi-frame confirmation.

4.3 Zero-Shot Advantages and Caveats

Zero-shot models facilitate new labels and cross-lingual scenarios without fine-tuning. However, performance depends on label phrasing, domain similarity, and multilingual coverage. Per-label thresholds and simple calibration (e.g., temperature scaling) substantially improve stability.

4.4 Real-Time Responsiveness

ROI scheduling, frame skipping, and micro-batching text for inference are reliable ways to maintain responsiveness. GPU usage, where available, helps transformers; CPU-only deployments benefit from quantized models and tokenizer reuse.

5. Discussion

5.1 Emerging Trends

- OCR robustness is foundational. Scene text detection plus strong sequence recognition is steadily becoming standard for noisy or cluttered scenes.

- Hybrid moderation is the norm. Stacking lexicon hits with transformer scores (and zero-shot labels) yields better coverage and resilience to obfuscation while allowing guardrails such as cooldowns.

- Tooling maturity matters. Standardized libraries for capture and overlays (MSS, layered windows) reduce engineering risk, while well-documented model hubs streamline deployment.

5.2 Conflicting Results and Reconciliation

- Precision vs. recall: Lexicon filters are precise yet brittle; transformers are more inclusive but can over-flag. This points to per-label thresholding and hybrid confirmation.

- Language and domain shifts: English-only models degrade on code-mixed Hindi–English. Multilingual checkpoints and small curated evaluation sets aligned with the deployment domain narrow the gap.

5.3 Implications for Engineering

- Add a text detector if OCR misses are high on cluttered UIs.

- Normalize aggressively post-OCR (Unicode normalization, simple transliteration mapping, digit/letter lookalike replacements) to recover lexicon hits.

- Adopt per-label thresholds and apply cooldowns to minimize alert fatigue.

- Implement an evaluation loop (see Section 6) and periodically recalibrate thresholds using new data.

6. Evaluation Blueprint

6.1 Datasets

- Internal samples: Curate consented screenshots representative of target UIs and languages. Ensure coverage of code-mixed Hindi–English and stylized fonts.

- Public resources: Use multilingual datasets (e.g., HASOC) for reference baselines while noting domain mismatch [11].

6.2 Metrics

- OCR: Character error rate (CER), word accuracy, script-specific breakdowns (Hindi vs. English).

- Classification: Precision/recall/F1 per label, ROC-AUC and PR-AUC, and expected calibration error (ECE) for probability outputs.

- System: End-to-end latency (capture → decision), frames per second, CPU/GPU utilization and memory footprint.

- Fairness: Group-wise false positive/negative rates across language and identity slices, parity/equalized odds analyses [9], [10].

6.3 Experimental Protocols

- Ablations: No detector vs. EAST-before-OCR; lexicon-only vs. transformer-only vs. hybrid; zero-shot label phrasing variants.

- Threshold sweeps: Construct precision–recall curves for each label; select operating points based on application risk tolerance.

- Robustness: Stress-test font changes, scaling, low-contrast text, translucency, and background motion reflective of modern UIs.

7. Ethical, Privacy, and Safety Considerations

- On-device inference: Prefer local processing to protect user data; if screenshots must be persisted, limit retention and apply redactions.

- Transparency and control: Provide notifications, explain detections, and allow users to override or whitelist cases.

- Bias auditing: Track disparities across language and identity slices; publish summaries and incorporate feedback loops for continuous improvement.

- Responsible defaults: Calibrate thresholds conservatively for sensitive categories and log uncertain cases for offline review rather than immediate enforcement.

8. Limitations and Future Work

- Code-mixed robustness: Develop OCR post-correction tailored to Hindi–English mixes; evaluate multilingual adapters or small-scale fine-tuning on representative samples.

- Calibration and bias: Use temperature scaling or isotonic regression; conduct periodic fairness audits with updated evaluation slices.

- Efficiency: Explore distilled/quantized transformers; cache tokenization and consider micro-batching across frames; use GPU where available.

- OCR–moderation co-design: Investigate joint training (or cascaded learning) where OCR outputs are refined toward moderation-aware vocabularies.

9. Conclusion

A reliable on-screen moderation stack emerges from the confluence of robust OCR, context-aware classification, and careful system engineering. The literature indicates that detector-enhanced OCR pipelines materially reduce recognition errors on cluttered UIs, while hybrid moderation—fusing lexicon cues with transformer and zero-shot signals—improves coverage and resilience to obfuscation. Real-time responsiveness is achieved through ROI scheduling, frame skipping, and hardware-aware inference. Finally, privacy and fairness are not afterthoughts: calibration, bias auditing, and user controls are essential for responsible deployment. Continued progress on code-mixed language handling, compact multilingual models, and moderation-aware OCR refinement will further stabilize real-world systems.

Acknowledgment

We thank the open-source communities and maintainers whose tools and documentation made this work possible, including the contributors to Tesseract OCR, the Hugging Face Transformers ecosystem, Kivy/KivyMD, and Python MSS, as well as the authors of platform guidance and APIs documented by Microsoft. We are grateful to the organizers and curators of multilingual toxicity resources (e.g., HASOC) for providing benchmarks that inform research on Hindi and code-mixed content.

We also appreciate the feedback and practical insights from colleagues and early testers who helped refine design choices for region-of-interest scheduling, threshold calibration, and user-centric notifications. Their input improved the clarity of our evaluation blueprint and strengthened our emphasis on fairness, privacy, and real-time responsiveness.

This work did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. Any remaining errors are our own.

References

[1] R. Smith, “An Overview of the Tesseract OCR Engine,” in Proc. ICDAR, 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4376991/

[2] Tesseract OCR, “4.0 with LSTM,” GitHub Docs. Accessed: Oct. 9, 2025. [Online]. Available: https://github.com/tesseract-ocr/tesseract/wiki/4.0-with-lstm

[3] X. Zhou et al., “EAST: An Efficient and Accurate Scene Text Detector,” in Proc. CVPR, 2017. [Online]. Available:

https://openaccess.thecvf.com/content_cvpr_2017/papers/Zhou_EAST_An_Efficient_CVPR_2017_paper.pdf

[4] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," arXiv:1507.05717, 2015. [Online]. Available: https://arxiv.org/abs/1507.05717

[5] Unitary AI, "unitary/toxic-bert," Hugging Face model card. Accessed: Oct. 9, 2025. [Online]. Available: https://huggingface.co/unitary/toxic-bert

[6] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing," arXiv:2111.09543, 2021. [Online]. Available: https://arxiv.org/abs/2111.09543

[7] A. Conneau et al., "XNLI: Evaluating Cross-lingual Sentence Representations," in Proc. EMNLP, 2018. [Online]. Available: https://aclanthology.org/D18-1269/

[8] Hugging Face, "What is Zero-Shot Classification?" Docs. Accessed: Oct. 9, 2025. [Online]. Available: https://huggingface.co/tasks/zero-shot-classification

[9] P. Fortuna and S. Nunes, "A Survey on Automatic Detection of Hate Speech in Text," ACM Computing Surveys, 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3232676

[10] A. Schmidt and M. Wiegand, "A Survey on Hate Speech Detection using NLP," in Proc. SocialNLP@EACL, 2017. [Online]. Available: https://aclanthology.org/W17-1101/

[11] HASOC, "HASOC 2019 Dataset," FIRE Shared Task. Accessed: Oct. 9, 2025. [Online]. Available: https://hasocfire.github.io/hasoc/2019/dataset.html

[12] Python MSS, "Welcome to Python MSS's documentation!" Accessed: Oct. 9, 2025. [Online]. Available: https://python-mss.readthedocs.io/

[13] Microsoft, "UpdateLayeredWindow function (winuser.h)," Learn Docs. Accessed: Oct. 9, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-updatelayeredwindow

[14] KivyMD, "Getting Started," Documentation. Accessed: Oct. 9, 2025. [Online]. Available: https://kivymd.readthedocs.io/en/latest/getting-started/