WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Context-Driven Optical Character Recognition Error Detection Framework for Secure Government Document Digitization

Akash Verma[1], Dr. Krishna Prasad R[2], Anish Raj B[3], Arpitha S Patil[4]

[1] *UG Student, Dept. of CSE Jyothy Institute of Technology, Bengaluru, India*
vermaakash6372@gmail.com
[2] *Associate Professor, Dept. of CSE Jyothy Institute of Technology, Bengaluru, India*
krishna.prasad@jyothyit.ac.in
[3] *UG Student, Dept. of CSE Jyothy Institute of Technology, Bengaluru, India*
anishrajb21@gmail.com
[4] *UG Student, Dept. of CSE Jyothy Institute of Technology, Bengaluru, India*
sarpitha604@gmail.com

ABSTRACT—

Government offices rely on Optical Character Recognition (OCR) to digitize identity cards, certificates and mark sheets, but OCR output from real scans is rarely error-free, and the mistakes travel quietly into verification and administrative databases. The hardest errors to catch are real-word errors, where a date or an ID number looks valid but is wrong. A spell-checker will not notice them. We built a web-based system that finds and helps correct such errors in scanned government documents. The scan is cleaned with OpenCV and read with Tesseract, and every word is tied to the field label it belongs to. Each word is then checked in three ways: format rules for structured fields like Aadhaar and PAN numbers, PIN codes and dates; a dictionary with edit-distance matching for misspellings; and a BERT or MuRIL masked language model that scores how well the word fits its sentence. These checks are combined with the OCR confidence into one error score, and words above a threshold get ranked suggestions. Nothing is changed until a reviewer approves it. Documents are encrypted, access is controlled through JWT and user roles, and every decision is logged. On a low-quality certificate scan, the system narrowed 55 recognised words down to 3 for review, one of them a wrong date, and all 52 automated tests passed.

***Index Terms*—OCR post-correction, error detection, BERT, MuRIL, masked language model, human-in-the-loop, document digitization, Indian languages, e-governance.**

I. Introduction

Across India, government departments are turning paper records into digital ones. Aadhaar and PAN cards, passports, residence and income certificates, mark sheets and land records are all being scanned and converted into searchable text, usually with an OCR engine such as Tesseract [1]. The scans themselves are often poor. Low resolution, rubber stamps, mixed fonts, faded ink, tilted pages and crowded layouts all cause recognition mistakes.

Many of these mistakes are familiar. The letter “O” becomes the digit “0”, “l” and “I” become “1”, and “S” becomes “5”. When a mistake produces a non-word, a dictionary can catch it [2]. Real-word errors are different: the output is a valid word, number or date that is simply wrong for that place in the document [2], [3]. On one of our test scans, a date of issue printed as “02/11/2025” came out as “02/14/2025”. Digitized records feed eligibility checks, identity verification and legal processes, so an error like this can do real damage.

Current tools only solve part of the problem. An OCR engine gives text but does not say which words to doubt. Spell-checkers skip numbers and IDs. Large language models can rewrite text confidently and still get it wrong [16]. In this paper we describe a system that brings these pieces together. It checks every OCR word with format rules, a dictionary with edit distance, and a contextual score from BERT [12] or MuRIL [13], and merges these with the OCR confidence into one error score. A reviewer must approve every correction, and the documents are kept encrypted with role-based access and an audit log. The system runs offline, supports English and 13 Indian languages, and can export the corrected document in its original layout.

Section II covers related work and Section III the system architecture. Section IV explains how errors are detected, Section V describes the implementation, Section VI presents results, and Section VII concludes the paper.

II. Related Work

Kukich [2] surveyed early spelling-correction research and separated non-word detection, isolated-word correction and context-dependent correction,

noting that the last was largely unsolved. Nguyen *et al.* [3] reviewed post-OCR processing approaches ranging from dictionaries and statistical models to machine translation and neural networks. The ICDAR 2017 [4] and ICDAR 2019 [5] post-OCR competitions released aligned OCR and ground-truth corpora and established error detection and error correction as separate tasks; in 2019 the strongest detection systems used BERT. However, these benchmarks consist of historical newspapers and books and contain no Indian-language forms or identity documents.

Kissos and Dershowitz [6] generated candidates from character confusion patterns and ranked them with a classifier that used OCR confidence and n-gram features. Bassil and Alwani [7] relied on an online spelling-suggestion service, which is unsuitable for confidential data that must remain offline. Mei *et al.* [8] learned ranking features from large corpora, while Amrhein and Clematide [9] treated post-correction as character-level statistical and neural machine translation, which requires large amounts of aligned training data. Schaefer and Neudecker [10] proposed a two-step approach that first detects erroneous tokens and then corrects only those, and Rijhwani *et al.* [11] addressed post-correction for low-resource languages.

Pre-trained transformers changed contextual error detection. BERT [12] predicts masked words from both directions of context, and MuRIL [13] extends this to 16 Indian languages and their transliterations. Hemmer *et al.* [14] injected OCR confidence into BERT embeddings for confidence-aware error detection. More recently, large language models have been prompted for OCR correction [15], but Kanerva *et al.* [16] found that these models sometimes make changes that read well but are wrong. Most of this work targets historical books and newspapers, relies on online services, or applies corrections with no one checking them. We did not find a system that combines field-level validation, contextual scoring for Indian languages, mandatory human review and secure storage, and that is what our work tries to provide.

III. System Architecture

Fig. 2 shows the five layers of the system: the React interface, the FastAPI service, the processing pipeline, the error detection and correction layer, and the data and security layer. Fig. 1 follows a single document through the system. A user signs in, picks the document type and OCR language, and uploads a scan. The image is cleaned, read and split into fields. Each word is scored by three detectors and the scores are combined. Words that look wrong go to a reviewer, and the reviewer's decisions are saved and used to produce the final output.

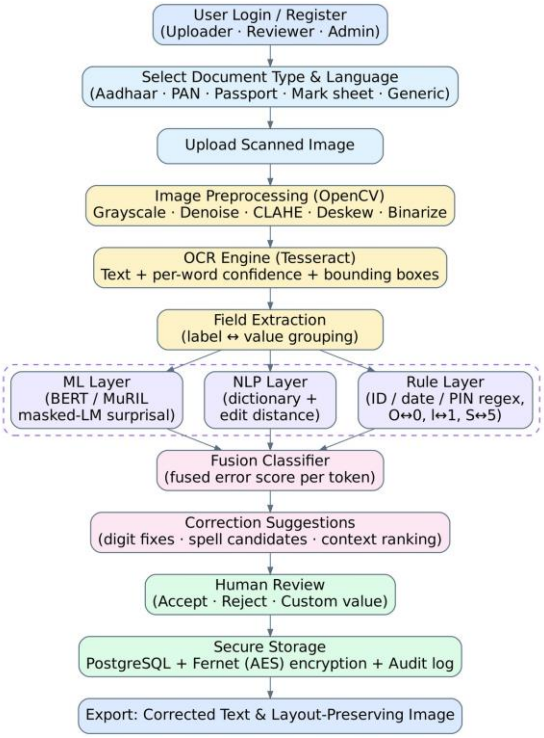


Fig. 1. End-to-end processing flow of the proposed system.

A. Image Preprocessing

Preprocessing reduces the noise that causes recognition errors. Using OpenCV [19], the image is converted to grayscale, cleaned with non-local means denoising [20], and enhanced with Contrast Limited Adaptive Histogram Equalization (CLAHE) [21] so that faded strokes become legible without amplifying background noise. The skew angle is estimated from the minimum-area rectangle enclosing the dark pixels and the page is rotated upright. Finally, adaptive Gaussian thresholding [22] binarizes the page while tolerating uneven illumination. The original image dimensions are retained so that corrections can later be drawn at their exact positions.

B. OCR and Field Extraction

Tesseract [1] is run in page-segmentation mode 6 and returns, for every word, its text, confidence (0–100), bounding box and block, paragraph and line indices. The requested languages are validated against the installed language data. Words are grouped into lines, and line prefixes are fuzzy-matched

against known labels such as *Name*, *Date of Birth*, *ID Number* and *Address*. From the label and the selected document type, each value is given a field kind such as date, Aadhaar, PAN, PIN code, passport, gender, name or address. Knowing the field lets the later checks compare a word with what should appear there.

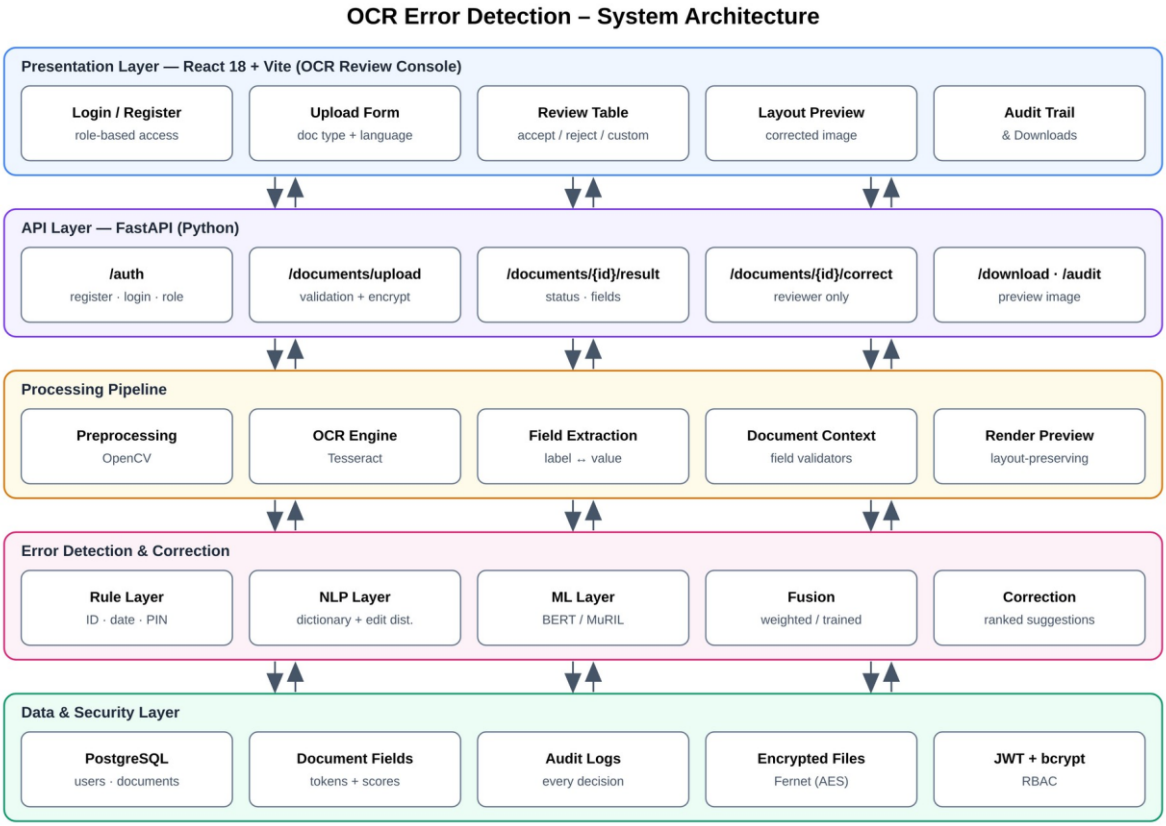


Fig. 2. Layered architecture of the system.

IV. Error Detection Methodology

Every token *w* passes through three separate detectors. Between them they handle structured fields, misspellings and real-word errors.

A. Rule-Based Validation

For IDs, numbers and dates, fixed rules work better than a language model. Aadhaar numbers must contain exactly 12 digits, PAN numbers must match five letters, four digits and one letter, and PIN codes must contain six digits. Dates are first matched with a regular expression and then checked as real calendar dates, so that impossible values such as month 14 or 31 February are rejected. Numeric tokens containing look-alike characters (O, o, l, I, |, S, B) are flagged as likely confusions, and a deterministic correction is generated by mapping each look-alike to its digit. The detector produces a binary rule flag $r \in \{0, 1\}$.

B. Dictionary and Edit-Distance Check

Alphabetic English words longer than two characters are looked up in a dictionary. Words that are not found are flagged, and spelling candidates within a small Levenshtein–Damerau edit distance [17], [18] are generated. Upper-case abbreviations, native-script words and structured fields are skipped to avoid false alarms on names and codes. This detector produces a binary flag $n \in \{0, 1\}$.

C. Contextual Masked-Language-Model Surprisal

To detect real-word errors, each eligible word is replaced by a [MASK] token inside a window C_w of up to 64 surrounding words, and a masked language model estimates the probability of the original word at that position. BERT (bert-base-uncased) [12] is used for English and MuRIL [13] for Indian and mixed-language documents. The contextual error signal is the surprisal

$$s(w) = -\log P(w \mid C_w) \qquad (1)$$

where a high value indicates that the word does not fit its context. All masked positions are scored in batches through the Hugging Face Transformers library [23] rather than with one forward pass per word. If the model cannot be loaded, the pipeline continues with the remaining signals.

D. Signal Fusion

The OCR confidence c, the two flags and the surprisal are combined into a single error score:

$$E(w) = 0.35(1 - c/100) + 0.25r + 0.15n + 0.25(1 - e^{-s}) \quad (2)$$

The term $1 - e^{-s}$ maps the unbounded surprisal into [0, 1). The weights reflect the reliability of each signal: low OCR confidence is the most general indicator, while rule violations and high surprisal are strong but specific evidence. When a trained logistic-regression model [24] is available, its predicted probability replaces the heuristic score. Tokens with $E(w) \geq 0.4$ are flagged.

E. Correction Suggestions and Human Review

For flagged numeric tokens the digit-mapped rule fix is proposed; for words, dictionary candidates and the masked model's top predictions are merged and ranked, with sub-word fragments removed. Suggestions are never applied automatically. A reviewer accepts a suggestion, rejects it to keep the original, or types a custom value. The backend verifies the reviewer's role, rebuilds the corrected text from token offsets, and records every approval and rejection in the audit log with the user and timestamp. We chose not to automate this step. One wrong digit can link a record to the wrong person, so a named reviewer should be responsible for each change.

V. Implementation

The frontend is built with React 18 and Vite and provides a three-step Upload–Review–Export workflow with keyboard navigation, screen-reader labels and light and dark themes. The backend uses Python 3.11 and FastAPI, runs the processing pipeline as a background task, and exposes REST endpoints for authentication, upload, status, results, corrections, previews, downloads and audit logs. PostgreSQL 16, accessed through SQLAlchemy, stores four entities: users, documents, document fields and audit logs. The document-fields table holds each token with its confidence, rule and dictionary flags, surprisal, fused score, suggestions, accepted correction and bounding box, allowing the corrected document to be reconstructed in its original layout.

Since the documents contain personal data, security was built in from the start. Passwords are hashed with bcrypt and sessions use HS256-signed JSON Web Tokens [25]. Every endpoint checks the user's role (uploader, reviewer or admin), so an uploader can neither approve corrections nor open another user's documents. Uploads are limited to 15 MB, checked for a valid image signature, and encrypted at rest with Fernet (AES). Stored file names derive only from a server-generated document identifier, preventing path-traversal attacks, and the temporary plaintext copy is deleted after processing. The system runs fully offline with no external API keys and is deployed with Docker Compose using CPU-only PyTorch. The layout-preview module draws approved corrections over the original scan at each word's bounding box and suppresses overlapping duplicate OCR boxes, and the result can be downloaded as a PNG image or UTF-8 text.

VI. Results and Discussion

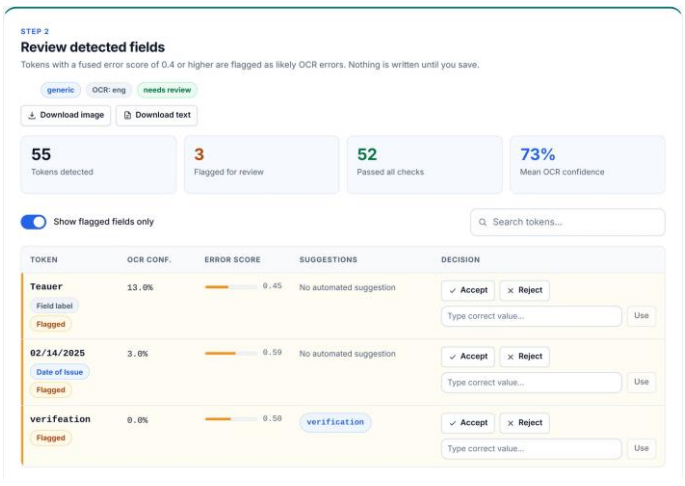
A. Case Study on a Degraded Scan

We tested the full workflow on a sample residence certificate filled with made-up details. It was scanned at low resolution to imitate a poor-quality government copy and processed in English. OCR returned 55 tokens with an average confidence of 73%. Only 3 were flagged and the other 52 passed, so the reviewer had to look at less than 6% of the text. Table I and Fig. 3 summarise the flagged tokens.

Table I. Tokens Flagged in the Case Study

OCR token	OCR conf.	Score E	Detection and outcome
02/14/2025	3.0%	0.59	Real-word date error under *Date of Issue*; rule layer rejected month 14. Reviewer entered 02/11/2025.
verifeation	0.0%	0.50	Non-word; dictionary suggested "verification", which the reviewer accepted.
Teauer	13.0%	0.45	Garbled field label flagged by very low OCR confidence. Reviewer entered "Issue:".

Fig. 3. Review screen showing the three flagged tokens with OCR confidence, fused error score and suggestions.



The date is the most useful catch. “02/14/2025” is a properly formatted date and would pass any spell-checker. Because it sat under the *Date of Issue* label, the rule layer checked it as a date and rejected it, since there is no 14th month. After the reviewer’s decisions were saved, the layout-preserving renderer replaced only the three approved words on the original scan (Fig. 4), and the audit trail recorded the upload, the automated processing and each approval with its timestamp.



Fig. 4. Layout-preserving preview with approved corrections drawn at their original positions.

B. Automated Testing

The rule, fusion, field-extraction and batching logic was tested with pytest, and the API was tested with FastAPI’s TestClient against a dedicated PostgreSQL database. All 52 automated test cases passed. Table II lists representative rule-layer cases.

Table II. Representative Rule-Layer Test Cases

Input	Field	Result
1234 5678 9012	Aadhaar	Accepted (12 digits)
1234 5678	Aadhaar	Rejected (too short)
12O456	Numeric	Flagged (O/0 confusion)
15/08/1998	Date of Birth	Not flagged
31/02/2020	Date of Birth	Rejected (impossible date)

Integration and system tests confirmed that uploads are validated, encrypted and logged; that an uploader is denied access to another user’s document; that a malicious file name cannot affect the stored path; and that OCR, rules, dictionary checks and fusion continue to work when the language model is unavailable. Testing also led to several fixes, including a rewritten date check with calendar validation, deterministic digit-level suggestions for ID numbers, suppression of overlapping OCR boxes in the preview, and batched model inference for long documents.

C. Discussion and Limitations

Each detector caught something the others would have missed. The rules found the wrong date, which the dictionary could not see. The dictionary fixed the misspelt word. The garbled label was caught only because its OCR confidence was very low, and no layer could suggest a fix for it, so the reviewer typed the correct value. Because review is mandatory, what was stored is what the reviewer approved, not the system’s guess.

Our evaluation has clear limits. It is qualitative and based on one English document. The fusion weights were set by hand rather than learned, and multi-page documents are not supported yet. The most important next step is to measure precision and recall on a labelled set of real government forms.

VII. Conclusion and Future Work

We presented a system for detecting OCR errors in government documents. It combines image preprocessing, Tesseract OCR, field-level rules, dictionary checks and BERT/MuRIL contextual scores into one error score for each word. The system flags only the words most likely to be wrong and asks a reviewer to decide on each one, so people spend their time where it is needed and every change is encrypted and logged. In our case study, 55 words were narrowed to 3 for review, including a wrong date that a spell-checker would have passed.

Future work includes training the fusion classifier on labelled data such as the ICDAR post-OCR benchmarks [4], [5] and real government forms; fine-tuning MuRIL on Indian-language documents; adding document-specific templates for Aadhaar, PAN, passports and mark sheets; handling merged and split words with sequence-to-sequence models such as TrOCR [26]; supporting multi-page PDFs and batch uploads; and building an active-learning loop in which reviewer decisions continuously improve the models.

Acknowledgment

The authors thank the management, the Principal, the Head of the Department and the faculty of the Department of Computer Science and Engineering, Jyothy Institute of Technology, Bengaluru, for their support and guidance throughout this work.

REFERENCES :

[1] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, 2007, pp. 629–633.

[2] K. Kukich, "Techniques for automatically correcting words in text," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377–439, 1992.

[3] T. T. H. Nguyen, A. Jatowt, M. Coustaty and A. Doucet, "Survey of post-OCR processing approaches," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–37, 2021.

[4] G. Chiron, A. Doucet, M. Coustaty and J.-P. Moreux, "ICDAR2017 competition on post-OCR text correction," in *Proc. 14th IAPR ICDAR*, 2017, pp. 1423–1428.

[5] C. Rigaud, A. Doucet, M. Coustaty and J.-P. Moreux, "ICDAR 2019 competition on post-OCR text correction," in *Proc. ICDAR*, 2019, pp. 1588–1593.

[6] I. Kissos and N. Dershowitz, "OCR error correction using character correction and feature-based word classification," in *Proc. 12th IAPR Workshop on Document Analysis Systems (DAS)*, 2016, pp. 198–203.

[7] Y. Bassil and M. Alwani, "OCR post-processing error correction algorithm using Google's online spelling suggestion," *J. Emerging Trends in Computing and Information Sciences*, vol. 3, no. 1, 2012.

[8] J. Mei, A. Islam, A. Moh'd, Y. Wu and E. Milios, "Statistical learning for OCR text correction," arXiv:1611.06950, 2016.

[9] C. Amrhein and S. Clematide, "Supervised OCR error detection and correction using statistical and neural machine translation methods," *J. Language Technology and Computational Linguistics*, vol. 33, no. 1, pp. 49–76, 2018.

[10] R. Schaefer and C. Neudecker, "A two-step approach for automatic OCR post-correction," in *Proc. LaTeCH-CLfL Workshop*, 2020, pp. 52–57.

[11] S. Rijhwani, A. Anastasopoulos and G. Neubig, "OCR post correction for endangered language texts," in *Proc. EMNLP*, 2020, pp. 5931–5942.

[12] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[13] S. Khanuja *et al.*, "MuRIL: Multilingual representations for Indian languages," arXiv:2103.10730, 2021.

[14] A. Hemmer, M. Coustaty, N. Bartolo and J.-M. Ogier, "Confidence-aware document OCR error detection," in *Proc. Document Analysis Systems (DAS)*, 2024, arXiv:2409.04117.

[15] J. Bourne, "CLOCR-C: Context leveraging OCR correction with pre-trained language models," arXiv:2408.17428, 2024.

[16] J. Kanerva, C. Ledins, S. Käpyaho and F. Ginter, "OCR error post-correction with LLMs in historical documents: No free lunches," in *Proc. NoDaLiDa/Baltic-HLT*, 2025, arXiv:2502.01205.

[17] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[18] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[19] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[20] A. Buades, B. Coll and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE CVPR*, 2005, pp. 60–65.

[21] K. Zuiderveld, "Contrast limited adaptive histogram equalization," in *Graphics Gems IV*, Academic Press, 1994, pp. 474–485.

[22] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[23] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proc. EMNLP: System Demonstrations*, 2020, pp. 38–45.

[24] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, IETF, 2015.

[26] M. Li *et al.*, "TrOCR: Transformer-based optical character recognition with pre-trained models," in *Proc. AAAI Conf. Artificial Intelligence*, 2023.