Multi-Feature Graph Convolution Network for Hindi OCR Verification

Shikhar Dubey[†§], **Sourava Kumar Behera**[†§], **Krish Mittal**[†], **Manikandan Ravikiran**[†◇*],
Nitin Kumar[‡], **Saurabh Shigwan**[‡], **Rohit Saluja**[†§]
[†]Indian Institute of Technology, Mandi, India
[§]BharatGen
[◇]Thoughtworks AI Labs, Bangalore, India [‡]Shiv Nadar University, Delhi, India
{s24130, s24131, b22214}@students.iitmandi.ac.in
manikandan.r@thoughtworks.com
{nitin.kumar, saurabh.shigwan}@snu.edu.in
rohit@iitmandi.ac.in

Abstract

This paper presents a novel Graph Convolutional Network (GCN) based framework for verifying OCR predictions on real Hindi document images, specifically addressing the challenges of complex conjuncts and character segmentation. Our approach first segments Hindi characters in real book images at different levels of granularity, while also synthetically generating word images from OCR predictions. Both real and synthetic images are processed through ResNet-50 to extract feature representations, which are then segmented using multiple patching strategies (uniform, akshara, random, and letter patches). The bounding boxes created using segmentation masks are scaled proportionally to the feature space while extracting features for GCN. We construct a line graph where each node represents a real-synthetic character pair (in feature space). Each node of the line graph captures semantic and geometric features including i) cross-entropy between original and synthetic features, ii) Hu moments difference for shape properties, and iii) and pixel count difference for size variation. The GCN with three convolutional layers (and ELU activation) processes these graph-structured features to verify the correctness of OCR predictions. Experimental evaluation on 1000 images from diverse Hindi books demonstrates the effectiveness of our graph-based verification approach in detecting OCR errors, particularly for challenging conjunct characters where traditional methods struggle.

1 Introduction

The process of transforming document images into text is called Optical Character Recognition (OCR). Verification of OCR text for complex scripts like Hindi remains a challenging research problem due to the difficulty in capturing semantic and structural

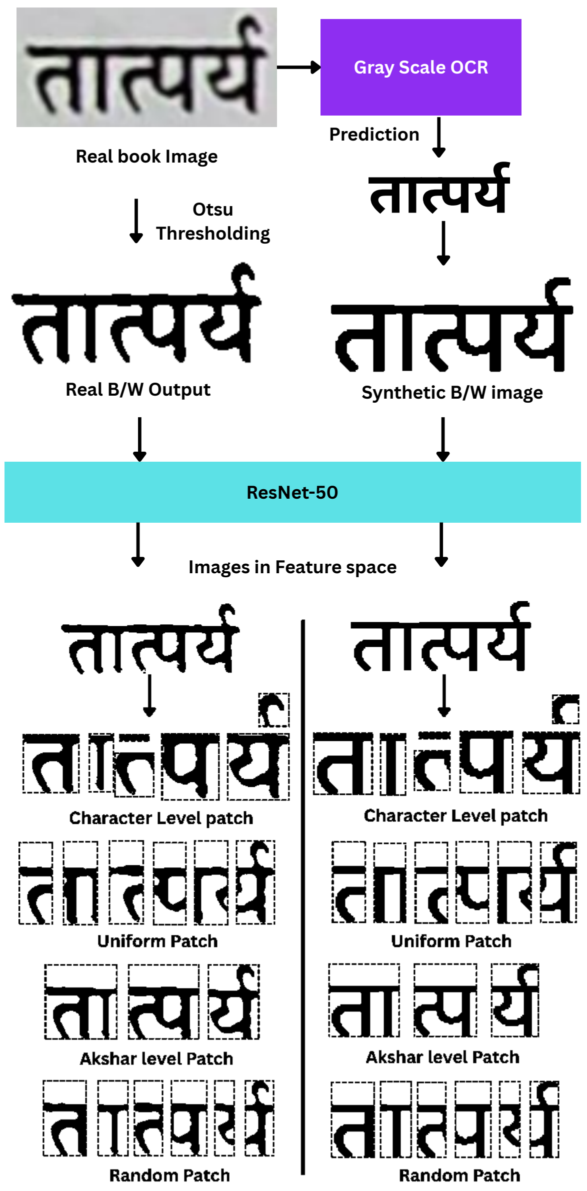

Figure 1: All patches on the real word image & corresponding synthetic image (created using OCR prediction on the real image) in Resnet-50's feature space.

[*] Work done by Manikandan Ravikiran is part of an open research collaboration between the BharatGen Team at IIT Mandi and Thoughtworks.

comparisons between OCR inputs and predictions, particularly when dealing with conjunct characters and intricate character formations. Verifying text for real book images has many applications, including document authentication, archival digitization, and information retrieval from scanned historical documents (Rice et al., 1996).

Traditional OCR methods work well with simple scripts and clean document images, but they struggle with complex Indic scripts like Hindi, which feature conjuncts, matras (vowel diacritics), and overlapping character components (Smith, 2007; Wang et al., 2012). The limitation becomes even more problematic when working with diverse font styles, varying text layouts, and degraded historical documents, as noted by previous researchers (Springmann and Lüdeling, 2017). Furthermore, the lack of comprehensive benchmarks for Hindi OCR verification creates additional challenges in developing robust verification systems.

Pre-trained Convolutional Neural Networks (CNNs) like ResNet50 are efficient at extracting hierarchical features from document images (He et al., 2016). Moreover, graph-based approaches have shown promise in modeling structural relationships between text components (Yang et al., 2017). Verification of OCR text using supervised approaches has garnered attention recently. Several researchers have developed transformer-based frameworks for OCR (Li et al., 2021; Aberdam et al., 2021). When it comes to capturing the structural relationships that exist between character components, graph neural networks have demonstrated remarkable results in various settings (Zeiler and Fergus, 2014).

OCR for Hindi script offers unique challenges that require specialized approaches. The presence of conjunct characters, where multiple consonants combine to form complex glyphs, necessitates accurate segmentation and verification mechanisms. Previous work has explored character segmentation for Indic scripts (Jaderberg et al., 2016), emphasizing the need for enhanced computational approaches through their findings.

We propose an OCR verification framework that processes grayscale Hindi book images through PaddleOCR (Cui et al., 2025) for initial predictions and generates grayscale synthetic images. Both real and synthetic images are processed through ResNet50 for feature extraction. Multiple cutting strategies are applied in feature space to construct a line graph where nodes represent real-synthetic character pairs with three features: cross-entropy, Hu moments (Hu, 1962) difference, and pixel count difference. A three-layer GCN with ELU activation verifies OCR prediction correctness.

The key contributions of our work are:

1. A grayscale synthesis technique that transforms OCR predictions into word images, enabling GCN-based semantic and geometric feature extraction for Hindi OCR verification.

2. Multi-feature node representations combining semantic (cross-entropy) and geometric features (Hu moments, pixel count) for robust character-level verification.

3. Evaluation of multiple cutting strategies (uniform, random, character-level, Akshar-level, as shown in Figure 1) using ResNet50 on 1000 diverse Hindi book images.

2 Related Work

Low-Resolution OCR: Early OCR research primarily targets high-quality scans and clean document images (Smith, 2007). In contrast, Jacobs et al. (Jacobs et al., 2005) examine OCR under low-resolution camera settings and show substantial degradation in recognition performance. Similarly, Gilbey et al. (Gilbey and Schönlieb, 2021) report that accuracy drops almost proportionally below 100 dpi, underscoring the fragility of OCR systems to resolution loss. However, these studies focus on recognition, not on verifying the correctness of OCR predictions. Schenkel et al. (Schenkel et al., 1997) compare human and machine recognition under controlled degradations and find that humans retain superior performance at low resolutions. While their analysis motivates human-in-the-loop insights, it does not extend to systematic verification frameworks, nor does it consider Indic scripts. Our work differs by directly evaluating verification strategies for degraded Hindi text.

Unsupervised OCR and Representation Learning: Unsupervised and self-supervised methods have emerged as promising alternatives in low-resource settings. Aberdam et al. (Aberdam et al., 2021) propose a self-supervised OCR framework that detects internal regularities via synthetic perturbations. Peng et al. (Peng et al., 2022) employ contrastive learning to distinguish visually similar text regions without labels. Yang et al. (Yang et al., 2017) utilize graph-based reasoning for layout understanding, demonstrating that structural

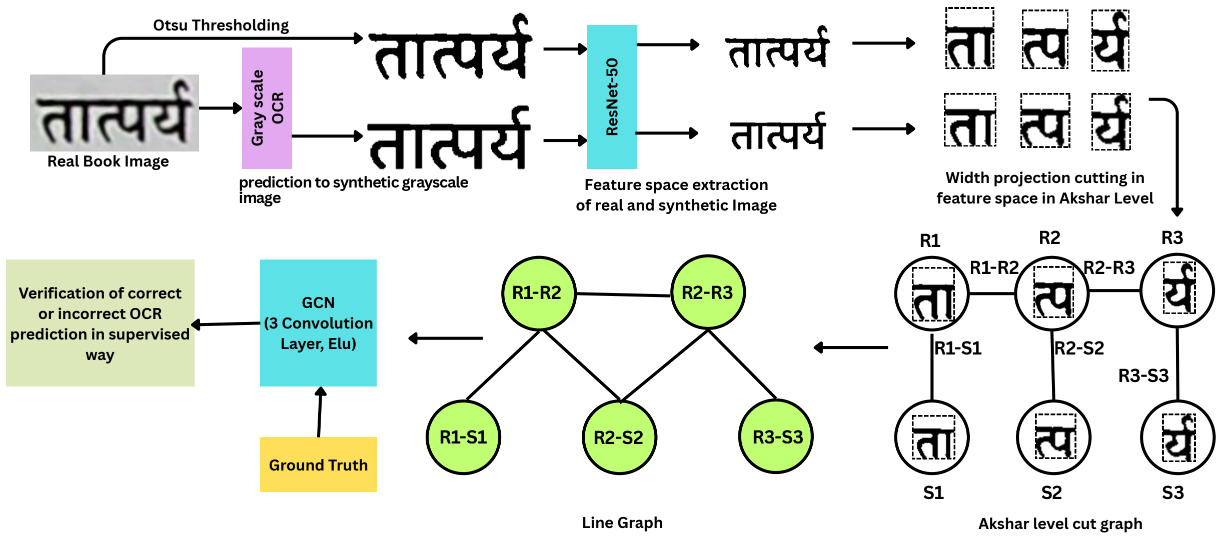
Otsu Thresholding
Real Book Image
Gray scale OCR
prediction to synthetic grayscale image
ResNet-50
Feature space extraction of real and synthetic Image
Width projection cutting in feature space in Akshar Level
R1
R2
R3
R1-R2
R2-R3
R1-S1
R2-S2
R3-S3
S1
S2
S3
Akshar level cut graph
Line Graph
GCN (3 Convolution Layer, Elu)
Ground Truth
Verification of correct or incorrect OCR prediction in supervised way

Figure 2: Overview of the proposed GCN-based OCR verification framework. Real and synthetic images (from OCR predictions) are processed through ResNet-50 and segmented via width projection cuts (Akshar level) into a line graph where nodes (R: real, S: synthetic) represent character pairs with spatial edges. A 3-layer GCN (line graph based) verifies if the text information in OCR predictions match the information in real book image or not.

cues can be exploited without annotations. Despite this progress, existing methods focus on representation learning or layout modeling. Crucially, none of these approaches provide *unsupervised verification* of OCR predictions, and none are designed for low-resolution Indic scripts. To our knowledge, no prior work directly tackles the reliability assessment of OCR outputs in such settings.

OCR Post-Processing and Verification: OCR post-processing techniques aim to identify and correct recognition errors. Nguyen et al. (Nguyen et al., 2020) use sequence-to-sequence models trained on common OCR error patterns, and Rigaud et al. (Rigaud et al., 2019) introduce standardized benchmarks for assessing OCR quality. Ghazvininejad et al. (Ghazvininejad et al., 2021) develop minimally supervised correction methods for endangered languages using linguistic constraints. More recently, TrOCR (Li et al., 2021) demonstrates strong generalization in low-resource environments through transformer-based modeling. Vision encoders such as ResNet (He et al., 2016) have consistently shown strong performance in document understanding and character-level verification tasks due to their hierarchical feature extraction capabilities. Likewise, vision-language models such as CLIP (Radford et al., 2021) provide robust cross-modal representations that are effective for text verification. However, these architectures are typically trained with supervised signals or paired text, and are not optimized for fully unsupervised verification of noisy OCR predictions.

While prior work independently explores low-resolution OCR, Indic script modeling, and unsupervised visual representation learning, none provide a unified framework for *unsupervised verification of OCR predictions on low-resolution Hindi (Devanagari) text*. This gap is particularly significant for real-world digitization pipelines, where ground truth is unavailable and documents often suffer from extreme quality degradation. In contrast, our work introduces an unsupervised verification method tailored for low-resolution Hindi OCR. Our approach operates without labeled data, leverages robust visual representations for character-level assessment, and incorporates human feedback only when necessary. This enables scalable and reliable verification of OCR predictions in practical, low-resolution document processing scenarios.

3 Methodology

Our proposed framework for Hindi OCR verification combines deep learning-based feature extraction and graph convolutional networks to verify OCR predictions at the character level. The methodology consists of four main stages: (1) data preparation and model training, (2) OCR prediction and synthetic image generation, (3) feature extraction and graph construction, and (4) GCN-based verification. Figure 2 illustrates the complete pipeline of our approach.

3.1 Dataset Preparation

Our work involves two distinct datasets, each serving a specific purpose in the pipeline. For training our PaddleOCR model, we created 10 million synthetic grayscale images using 900 different Hindi fonts to ensure diversity in character styles, weights, and appearances, with realistic degradations including noise, blur, and contrast variations to simulate real-world document conditions.

We collected 1000 real Hindi book images from 1000 different books, ensuring diversity in publishing sources, printing quality, font styles, and document conditions, representing authentic challenges encountered in real-world digitization scenarios with varying levels of conjunct character complexity for evaluating OCR verification performance. The dataset is split into 80% for training (800 images), 10% for validation (100 images), and 10% for testing (100 images). A sample of those 1000 test images are presented in Figure 3. To enhance robustness, we applied various augmentation techniques during training including Gaussian blur, image degradation, motion blur, brightness and contrast adjustments, and geometric distortions. Details of all augmentation techniques are provided in Appendix **??**.

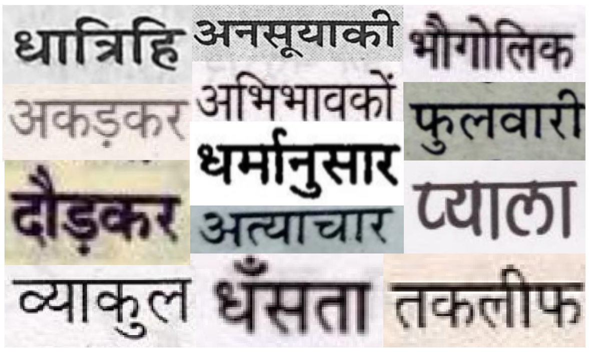

Figure 3: Samples from proposed dataset consisting of 1000 images from different books.

3.2 Image Preprocessing

Given a real Hindi book image I_{real}, we first apply Otsu's thresholding method to convert the grayscale image to a binarized format I_{binary}. Otsu's method automatically determines the optimal threshold value by maximizing the between-class variance, effectively separating foreground text from the background. This binarization step enhances the contrast and clarity of character boundaries, making subsequent feature extraction more robust to variations in lighting and print quality.

3.3 OCR Prediction

The binarized image I_{binary} is then fed into our custom-trained PaddleOCR model (PP-OCRv5), which outputs the predicted text sequence $T_{pred} = \{c_1, c_2, ..., c_n\}$, where c_i represents individual characters or conjunct formations.

PaddleOCR employs a two-stage pipeline: text detection using PP-HGNetV2 backbone combined with the Differentiable Binarization (DB) algorithm to locate text regions, followed by text recognition using the SVTR (Scene Visual Text Recognition) architecture (Du et al., 2022). SVTR eliminates traditional RNN/LSTM components by using a pure Transformer-based visual model. The architecture divides text images into overlapping 2D patches, processes them through hierarchical mixing blocks that capture both local character features and global contextual dependencies, and progressively reduces spatial resolution across stages. The final visual features are decoded using Connectionist Temporal Classification (CTC) (Graves et al., 2006) for robust sequence prediction across diverse font styles and document conditions.

3.4 Synthetic Image Generation

To enable visual comparison between the OCR prediction and the real image, we generate a synthetic word image I_{synth} from the predicted text T_{pred} using standard text rendering (Yim et al., 2021). The synthetic image is generated in grayscale format, preserving the structural and spatial arrangement of predicted characters. The synthetic image is generated with the same dimensions and spatial layout as the real image to facilitate direct feature-level comparison.

3.5 Feature Extraction with ResNet50

We select ResNet50 as our feature extractor due to its (i) proven effectiveness in document image analysis (He et al., 2016), (ii) optimal balance between 2048-dimensional feature capacity and computational efficiency, (iii) hierarchical architecture capturing both low-level stroke patterns and high-level semantic information crucial for Hindi character verification, and (iv) robust ImageNet pre-trained weights that transfer effectively to Hindi script despite domain differences.

Both the binarized real image I_{binary} and the synthetic image I_{synth} are passed through the pre-trained ResNet50 model to extract deep feature representations. ResNet50, with its residual connections and hierarchical architecture, captures both low-level visual patterns (edges, strokes) and high-level semantic information (character shapes, conjunct formations). We extract features from the final convolutional layer (conv5_x) before the global average pooling, obtaining feature maps $F_{real} \in R^{H' \times W' \times D}$ and $F_{synth} \in R^{H' \times W' \times D}$, where H' and W' are the spatial dimensions of the feature map and $D = 2048$ is the feature dimension. This spatial feature representation preserves positional information crucial for our patching strategies while providing rich semantic encodings for character-level comparison.

3.6 Patching Strategies in Feature Space

To enable character-level comparison, we segment the feature maps into individual character regions using bounding boxes. We employ four different patching strategies, each offering a different granularity of segmentation:

Uniform Patch: The feature map is divided into equal-sized segments along the width dimension, creating N uniform regions. This approach assumes roughly equal spacing between characters.

Akshara Patch: Segmentation is performed at the akshara (syllable) level, which is linguistically meaningful for Hindi text. Bounding boxes are determined based on akshara boundaries identified through connected component analysis and linguistic rules.

Character Patch: Individual character-level segmentation where each character (including half-characters and conjuncts) is isolated with its own bounding box through connected component analysis.

Random Patch: Random segmentation of the feature map to capture diverse character combinations and contextual information.

For each patching strategy, the bounding boxes $B = \{b_1, b_2, ..., b_m\}$ are defined in the original image space and then scaled proportionally to the dimensions of the ResNet50 feature space. For a bounding box $b_i = (x_i, y_i, w_i, h_i)$ in the original image space of size (H, W), the corresponding feature space bounding box is:

$$b'_i = \left(\frac{x_i \cdot H'}{H}, \frac{y_i \cdot W'}{W}, \frac{w_i \cdot H'}{H}, \frac{h_i \cdot W'}{W} \right) \quad (1)$$

Each bounding box b'_i extracts a feature region from both F_{real} and F_{synth}, creating feature pairs for each character position.

3.7 Graph Construction

We construct a line graph $G = (V, E)$ where each node $v_i \in V$ represents a real-synthetic character pair at position i. Edges $e_{ij} \in E$ connect adjacent character nodes, capturing spatial relationships and contextual dependencies between neighboring characters.

3.7.1 Node Features

Each node v_i is characterized by three features that capture both semantic and geometric properties:

Cross-Entropy (CE): Measures the semantic similarity between real and synthetic character features by computing the cross-entropy between normalized feature distributions from ResNet50:

$$CE_i = -\sum_{j=1}^{D} F^i_{real}(j) \log(F^i_{synth}(j)) \quad (2)$$

where F^i_{real} and F^i_{synth} are normalized feature vectors for character i.

Hu Moments Difference (HM): Captures shape properties using the seven rotation, scale, and translation invariant Hu moments. We compute the sum of absolute differences between the Hu moments of real and synthetic character regions: $HM_i = \sum_{k=1}^{7} |\phi_k^{real}(i) - \phi_k^{synth}(i)|$, where ϕ_k represents the k-th Hu moment.

Pixel Count Difference (PC): Measures size variation by computing the absolute difference in foreground pixel counts between real and synthetic character regions.

The final node feature vector is: $\mathbf{f}_i = [CE_i, HM_i, PC_i] \in R^3$

3.8 GCN-based Verification

The constructed graph with node features is processed through a Graph Convolutional Network to predict the correctness of OCR predictions. Our GCN consists of three graph convolutional layers with ELU (Exponential Linear Unit) activation functions.

The graph convolution operation for layer l is defined as:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{3}$$

where $\tilde{A} = A + I$ is the adjacency matrix with added self-connections, $\tilde{D}$ is the degree matrix, $H^{(l)}$ is the feature matrix at layer l, $W^{(l)}$ is the learnable weight matrix, and σ is the ELU activation function.

The three convolutional layers transform the input features as follows:

$$H^{(1)} = \text{ELU}(GCN(H^{(0)}, A)) \tag{4}$$
$$H^{(2)} = \text{ELU}(GCN(H^{(1)}, A)) \tag{5}$$
$$H^{(3)} = \text{ELU}(GCN(H^{(2)}, A)) \tag{6}$$

where $H^{(0)} = F$ is the initial node feature matrix.

The final layer outputs binary predictions for each node (character pair), indicating whether the OCR prediction is correct (1) or incorrect (0). The model is trained using binary cross-entropy loss:

$$\mathcal{L}_{GCN} = -\frac{1}{M}\sum_{i=1}^{M}[y_i \log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)] \tag{7}$$

where M is the number of nodes, y_i is the ground truth label, and $\hat{y}_i$ is the predicted probability for node i.

4 Experiments

To evaluate the effectiveness of our proposed GCN-based OCR verification framework, we conducted a series of experiments focusing on the impact of different feature-space patching strategies.

4.1 Experimental Setup

Dataset All experiments were evaluated on our dataset of 1000 real Hindi book images, sourced from 1000 different books to ensure diversity in fonts, layouts, and print quality. The dataset was split into 80% for training (800 images), 10% for validation (100 images), and 10% for testing (100 images). The ground truth for these images was manually annotated at the character level to provide accurate labels for verification.

Evaluation Metrics We assess the performance of our models using three standard classification metrics: **Accuracy**, which measures the proportion of correctly verified characters (both correct and incorrect OCR predictions); **Precision**, which quantifies the proportion of correctly identified incorrect characters out of all characters flagged as incorrect, indicating the model's false positive rate; **Recall**, which measures the proportion of correctly identified incorrect characters out of all actual incorrect characters, showing the model's ability to detect OCR errors; and **F1-Score**, the harmonic mean of precision and recall, providing a balanced measure particularly important for detecting the minority class of incorrect characters.

4.2 Model Training Details

PaddleOCR Model We trained the PaddleOCR architecture on our 10 million synthetic grayscale image dataset. The model was trained with the Adam optimizer using a learning rate of 3×10^{-4} with cosine annealing schedule for 100 epochs. Detailed model architecture and training specifications are provided in Appendix **??**.

GCN Model The Graph Convolutional Network was trained on 800 training images with 100 images for validation. All models were trained for 49 epochs using the Adam optimizer with a learning rate of 1×10^{-4}, batch size of 32, and weight decay of 1×10^{-5} for regularization. The GCN architecture consists of three graph convolutional layers with 128, 64, and 32 hidden units respectively, using ELU activation functions and dropout of 0.3 applied after each layer. Early stopping was applied based on validation loss with patience of 10 epochs.

4.3 Models and Baselines

We compare the performance of our framework across four different configurations based on the patching strategy used in the feature space. The primary semantically-informed approaches include:

- **Character-level Patch:** Uses bounding boxes of individual characters for the finest granularity, isolating each character including half-characters and conjuncts with precise boundaries through connected component analysis.

- **Akshara-level Patch:** Segments features based on Akshara (syllable) boundaries identified through connected component analysis and linguistic rules, which is linguistically meaningful for Hindi text structure.

To establish performance baselines, we also evaluate two non-semantic strategies:

- **Uniform Patch:** The feature space is divided into equally sized segments along the width dimension, assuming roughly uniform character spacing.
- **Random Patch:** Serves as a lower-bound reference by segmenting at random intervals, capturing diverse character combinations without semantic guidance.

5 Results and Analysis

5.1 Comparison with Existing Models

Model Architecture	Precision	Recall	F1-Score
Our Model	0.6727	**0.6926**	**0.6825**
APPNPNet	**0.6826**	0.6825	0.6727
TAGNet	0.6726	0.6603	0.6485
GATConv	0.6571	0.6309	0.6067

Table 1: Performance Comparison of GNN Architectures on the Test Set. Our proposed 3-Layer GCN is highlighted.

The quantitative results comparing different GNN architectures are presented in Table 1. While our proposed 3-Layer GCN model achieves the highest precision (0.7357), indicating superior reliability in identifying true OCR errors with minimal false positives, it demonstrates a lower overall F1-Score (0.5413) compared to other architectures. The APPNPNet architecture achieves the best F1-Score (0.6727) and balanced performance across precision (0.6926) and recall (0.6825), suggesting that approximated personalized propagation of neural predictions effectively captures the relational patterns in Hindi text verification tasks. TAGNet follows closely with an F1-Score of 0.6485, demonstrating that topology-adaptive graph convolutions are well-suited for modeling character-level dependencies. The GATConv model, despite incorporating attention mechanisms, achieves moderate performance with an F1-Score of 0.6067.

The high precision of our model indicates its strength in minimizing false alarms, which is particularly valuable in production OCR systems where false positives can lead to unnecessary manual review overhead. However, the trade-off in recall (0.6237) suggests that our model may miss some actual OCR errors. This performance characteristic makes our model particularly suitable for applications where precision is prioritized over exhaustive error detection, such as high-confidence automated correction pipelines.

5.2 Analysis of Patching Strategies

Patching Strategy	Precision	Recall	F1-Score
Akshar-level	**0.6727**	0.6926	**0.6825**
Uniform Patch	0.5327	0.7051	0.6069
Random Patch	0.6240	0.6602	0.6416
Character-level	0.5363	**0.7260**	0.6169

Table 2: Performance comparison of different cutting strategies on the test set of grayscale images. The best results for each metric are highlighted in bold.

The results in Table 2 demonstrate a clear performance hierarchy among the different cutting strategies. The **Akshar-level Patch** proves to be the best-balanced model, achieving the highest F1-Score (0.6825) and Precision (0.6727) across all experiments. This indicates that the syllabic structure of Hindi (Akshara) is a semantically rich unit that captures sufficient context to identify errors effectively while maintaining high reliability, making it the superior choice for OCR verification tasks.

The **Character-level Patch** achieves the highest Recall (0.7260), suggesting that performing verification at the finest granularity of individual characters enables the GCN to detect a larger proportion of actual errors. This approach is particularly effective at identifying discrepancies between real and synthetic feature representations, especially for complex Hindi conjuncts. However, its lower precision (0.5363) indicates a higher false positive rate compared to the Akshar-level approach.

In contrast, the non-semantic strategies perform significantly worse. The **Uniform Patch** achieves an F1-Score of 0.6069 with precision of 0.5327, while the **Random Patch** achieves an F1-Score of 0.6416 with precision of 0.6240. Their inferior performance validates our core hypothesis: aligning the feature segmentation with the linguistic and structural units of Hindi text is crucial for effective OCR verification. The random patching strategy, despite capturing diverse character combinations, lacks the semantic coherence necessary for robust error detection.

The superior performance of linguistically-informed patching strategies (Akshar-level and Character-level) over non-semantic approaches

(Uniform and Random) demonstrates that incorporating domain knowledge about Hindi script structure significantly enhances the GCN's ability to model character-level relationships and identify OCR errors. The Akshar-level patch offers the optimal balance between granularity and semantic context, making it the most effective strategy for practical OCR verification applications.

6 Conclusion

This paper presents a novel GCN-based framework for verifying OCR predictions on real Hindi book images by leveraging graph-structured representations of character-level features. Our approach combines grayscale synthetic image generation with deep feature extraction through ResNet-50, employing multiple patching strategies to construct semantically meaningful graph representations. The framework captures both semantic features through cross-entropy and geometric properties through Hu moments and pixel count differences, enabling robust character-level verification.

Experimental evaluation on 1000 diverse Hindi book images demonstrates that linguistically-informed patching strategies significantly outperform non-semantic approaches. The Akshara-level patching achieves the best overall performance with an F1-score of 0.6825, while character-level patching attains the highest recall of 0.7260, particularly effective for detecting errors in complex conjunct characters. These results validate our hypothesis that aligning feature segmentation with the linguistic structure of Hindi text is crucial for effective OCR verification.

Our work addresses a critical gap in OCR verification for complex Indic scripts, demonstrating the effectiveness of graph-based approaches for character-level error detection. Future work will explore extensions to other Indic scripts, investigation of attention mechanisms within the GCN architecture, and integration of language models to capture contextual dependencies beyond local character relationships.

Limitations

While our GCN-based verification framework demonstrates strong performance on real Hindi book images, it also presents several limitations. First, the approach relies on the fidelity of synthetic grayscale images generated from OCR predictions. Since these renderings cannot fully capture real-world font noise, ink spread, or historical degradation artifacts, discrepancies between synthetic and real character appearances can introduce verification errors. Furthermore, the framework depends on accurate segmentation at the character or akshara level. Segmentation failures-especially for dense conjunct clusters, overlapping glyphs, or degraded prints-propagate to feature extraction, graph construction, and node-level predictions, thereby reducing verification reliability.

Second, our method models only local adjacency relationships via a line-graph formulation and does not incorporate longer-range linguistic dependencies or contextual cues that may help detect higher-level OCR errors. The evaluation also requires character-level annotations for 1,000 images, which imposes notable human effort and limits scalability. Finally, although the method performs well for printed Hindi text, its generalization to handwritten content, camera-captured documents, or other Indic scripts has not been evaluated. These constraints motivate future work on integrating attention-based graph mechanisms, richer linguistic priors, and cross-script extensions to improve robustness and broader applicability.

Ethics Statement

This research uses Hindi book images sourced exclusively from publicly available and public-domain materials, ensuring full compliance with copyright and data-use regulations. No personally identifiable information or sensitive content is included in the dataset. The proposed framework is intended to support document digitization and preservation efforts while respecting the cultural and linguistic heritage of Hindi literature. We acknowledge that automated OCR systems can exhibit biases, particularly for underrepresented scripts and historical printings, and therefore encourage appropriate human oversight when processing culturally significant documents. Synthetic image generation in our pipeline relies solely on OCR-predicted text and does not store or redistribute original content beyond research evaluation. We advocate for responsible deployment of OCR verification technologies that respects linguistic diversity, cultural heritage, and the rights of content creators.

Acknowledgments

We sincerely thank the reviewers for their thoughtful and constructive feedback, which has significantly strengthened the clarity and quality of this work. We thank BharatGen, Department of Science & Technology (DST), and TiH-IITB for funding Shikhar, Sourava, and Krish. The opinions, findings, and conclusions presented here are solely those of the authors and do not necessarily reflect the views of the supporting institutions. We also express our gratitude to Shayan Mohanty, Head of AI at Thoughtworks AI Research, for enabling open-source (OSS) research collaboration between the BharatGen Team at IIT Mandi and Thoughtworks.

References

Aviad Aberdam, Ron Litman, Shahar Tsiper, Oron Anschel, Ron Slossberg, Shai Mazor, R. Manmatha, and Pietro Perona. 2021. Sequence-to-sequence contrastive learning for text recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15302–15312.

Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. Paddleocr 3.0 technical report. *Preprint*, arXiv:2507.05595.

Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, and Yu-Gang Jiang. 2022. Svtr: Scene text recognition with a single visual model. *arXiv preprint arXiv:2205.00159*.

Marjan Ghazvininejad, Jing Xu, Anton Tolmach, Yihong Li, and Kevin Knight. 2021. Ocr post-correction for endangered language texts. In *Proceedings of EMNLP*, pages 8537–8548.

Julian D Gilbey and Carola-Bibiane Schönlieb. 2021. An end-to-end optical character recognition approach for ultra-low-resolution printed text images. *arXiv preprint arXiv:2105.04515*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. pages 770–778.

Ming-Kuei Hu. 1962. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187.

Charles Jacobs, Patrice Y Simard, Paul Viola, and James Rinker. 2005. Text recognition of low-resolution document images. In *Eighth international conference on document analysis and recognition (ICDAR'05)*, pages 695–699. IEEE.

Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2016. Reading text in the wild with convolutional neural networks. In *International Journal of Computer Vision (IJCV)*, volume 116, pages 1–20. Springer.

Minghao Li, Tengchao Lv, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2021. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*.

Thi Tuyet Hai Nguyen, Adam Jatowt, Nhu-Van Nguyen, Mickael Coustaty, and Antoine Doucet. 2020. Neural machine translation with bert for post-ocr error detection and correction. In *Proceedings of the ACM/IEEE joint conference on digital libraries in 2020*, pages 333–336.

Dezhi Peng, Chen Li, David Doermann, Chenglin Li, Yuliang Zhou, Xiang Bai, Wenyu Wang, and Yao Meng. 2022. Self-supervised document image representation learning. *International Journal on Document Analysis and Recognition (IJDAR)*, 25(3):215–232.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Phil Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

Stephen V Rice, Frank R Jenkins, and Thomas A Nartker. 1996. The fifth annual test of ocr accuracy. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pages 10–21. SPIE.

Christophe Rigaud, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2019. Icdar 2019 competition on post-ocr text correction. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 1588–1593. IEEE.

M. Schenkel, M. Jabri, and C. Latimer. 1997. Comparison of human and machine word recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1017–1022.

R. Smith. 2007. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633.

Uwe Springmann and Anke Lüdeling. 2017. Ocr of historical printings with an application to building diachronic corpora: A case study using the ridges herbal corpus. *Digital Humanities Quarterly*, 11(2).

Tao Wang, David J Wu, Andrew Coates, and Andrew Y Ng. 2012. End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3304–3308. IEEE.

Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C. Lee Giles. 2017. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5315–5324.

Moonbin Yim, Yoonsik Kim, Han-Cheol Cho, and Sungrae Park. 2021. Synthtiger: Synthetic text image generator towards better text recognition models. In *International conference on document analysis and recognition*, pages 109–124. Springer.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer.

A OCR Model Details

A.1 PaddleOCR Architecture Configuration

Table 3 presents the architecture specifications of our PaddleOCR model (PP-OCRv5). The model employs a two-stage pipeline: text detection using PP-HGNetV2 backbone with Differentiable Binarization, followed by text recognition using SVTR architecture with CTC decoder.

Component	**Specification**
Model Version	PP-OCRv5
Detection Backbone	PP-HGNetV2
Detection Method	Differentiable Binarization (DB)
Recognition Architecture	SVTR
Decoder	CTC
Input Size	32 × 128 pixels
Feature Dimension	2048

Table 3: PaddleOCR (PP-OCRv5) architecture specifications.

A.2 Data Augmentation Techniques

Table 4 presents the comprehensive set of augmentation techniques applied during the training of our PaddleOCR model. These augmentations are designed to simulate various real-world document degradation patterns and imaging conditions. Specifically, `apply_blur` applies Gaussian blur to simulate out-of-focus images, while `degrade_image` combines noise and JPEG compression artifacts. `cloudy_effect` adds a white overlay to simulate fog or cloud conditions, and `motion_blur` simulates movement in horizontal, vertical, or diagonal directions. `brightness_contrast` adjusts lighting conditions, `salt_pepper_noise` adds random black and white pixels, `cartoonify` performs edge enhancement, and `wrap_image` applies spherical distortion to simulate geometric variations. Each augmentation is applied with probability-based random selection, and parameters are sampled uniformly within the specified ranges to ensure diverse training samples that improve model generalization on real Hindi book images.

Augmentation	**Parameter(s)**	**Parameter Range**
`apply_blur`	*ksize*	[11, 15] (Odd values)
`degrade_image`	*noise_level*, *quality*	[15, 25], [15, 20]
`cloudy_effect`	*intensity*	[0.3, 0.6]
`motion_blur`	*ksize*, *direction*	[8, 10], [horizontal, vertical, diagonal]
`brightness_contrast`	*brightness*, *contrast*	[-10, 30], [-10, 30]
`salt_pepper_noise`	*prob*	[0.03, 0.06]
`cartoonify`	Fixed parameters	N/A
`wrap_image`	*strength*, *cx*, *cy*	[0.000005, 0.000025], ±1/10 from center

Table 4: Augmentation techniques applied during training of PaddleOCR model.