GraDeT-HTR: A Resource-Efficient Bengali Handwritten Text Recognition System utilizing Grapheme-based Tokenizer and Decoder-only Transformer

Md. Mahmudul Hasan*, **Ahmed Nesar Tahsin Choudhury***,
Mahmudul Hasan, **Md. Mosaddek Khan**
Computer Science and Engineering, University of Dhaka
{mdmahmudul-2020215620, ahmednesartahsin-2020115612,
mahmudul-2019917803}@cs.du.ac.bd, mosaddek@du.ac.bd

Abstract

Despite Bengali being the sixth most spoken language in the world, handwritten text recognition (HTR) systems for Bengali remain severely underdeveloped. The complexity of Bengali script—featuring conjuncts, diacritics, and highly variable handwriting styles—combined with a scarcity of annotated datasets makes this task particularly challenging. We present **GraDeT-HTR**[1], a resource-efficient Bengali handwritten text recognition system based on a **Gra**pheme-aware **De**coder-only **T**ransformer architecture. To address the unique challenges of Bengali script, we augment the performance of a decoder-only transformer by integrating a grapheme-based tokenizer and demonstrate[2] that it significantly improves recognition accuracy compared to conventional subword tokenizers. Our model is pretrained on large-scale synthetic data and fine-tuned on real human-annotated samples, achieving state-of-the-art performance on multiple benchmark datasets.

1 Introduction

Optical Character Recognition (OCR) involves transforming printed or handwritten images into machine-encoded text, supporting tasks such as digitizing pages from books, scene photographs, receipts, or notes. Within OCR, Handwritten Text Recognition (HTR) is focused specifically on transcribing handwritten content. HTR remains a challenging task due to the vast diversity in individual writing styles, inconsistent lighting, varying stroke width, cursive scripts, and artifacts such as smudges or noise.

Despite significant progress in handwritten text recognition (HTR) for languages such as English and Chinese, Bengali remains notably underserved. This gap is especially critical given the widespread use of handwritten Bengali in education, administration, and historical records. Developing effective HTR systems for Bengali presents unique challenges: the script contains visually complex ligatures, numerous diacritics, and allographic variations, all of which are compounded by highly diverse individual handwriting styles. These linguistic and visual complexities, combined with a scarcity of large, high-quality annotated datasets, make Bengali HTR a particularly demanding low-resource OCR task.

Existing approaches to text recognition predominantly employ encoder-decoder architectures, wherein the encoder is responsible for extracting visual features from images while the decoder generates the corresponding transcriptions. Early methods (Shi et al., 2017) commonly utilized CNNs as encoders and RNNs, with CTC (Graves et al., 2006), as decoders. With the introduction of transformers (Vaswani et al., 2017), attention-based mechanisms have been utilized to gain high performance in OCR tasks (Wang et al., 2019; Sheng et al., 2019; Zhang et al., 2020; Li et al., 2023; Fujitake, 2024).

In the context of Bengali, existing research on text recognition is relatively limited. Recent studies have explored architectures based on Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). However, the majority of these works (Azad et al., 2020; Hossain et al., 2021; Safir et al., 2021; Chaudhury et al., 2022) evaluate their methods on specific datasets, resulting in limited generalizability of their results. Some recent efforts (Hossain et al., 2022) have considered evaluation on unseen datasets, thereby addressing issues of generalization to a certain extent. Nonetheless, approaches based on transformer architectures have not yet been explored for HTR tasks. For text detection, models such as BN-DRISHTI (Jubaer et al., 2023) attain satisfactory results, so the focus has largely shifted to the recognition stage. In paral-

*Equal contribution. Author order is randomly determined.
[1] https://cognistorm.ai/hcr
[2] https://github.com/mahmudulyeamim/GraDeT-HTR

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 696–706

lel, recent work on tokenizer design (Basher et al., 2023) has demonstrated that grapheme-based tokenization can improve recognition performance by addressing Bengali's intricate ligatures, conjuncts, numerous diacritics, and allographic variations.

Motivated by the strong performance of decoder-only transformer models in English and Chinese OCR tasks (Fujitake, 2024), as well as the potential significance of tokenizer selection, we propose a Bengali Handwritten Text Recognition system that leverages a decoder-only transformer architecture in conjunction with a grapheme-based tokenizer. The core contributions of this paper are summarized below:

- We develop an end-to-end Bengali HTR pipeline that integrates both text detection and recognition for full-page images.
- We study the effects of replacing the default tokenizer in a transformer-based model with a grapheme-based tokenizer tailored for Bengali—a direction that, to the best of our knowledge, has not been previously explored—and empirically show improvements in recognition performance.
- We demonstrate that strong HTR performance can be achieved without relying on large-scale pretrained language models, offering a practical path for developing this demanding application in low-resource settings.

2 Related Work

This section reviews prior research in three key areas relevant to our work: general optical character recognition (OCR) and handwritten text recognition (HTR), Bengali handwritten text recognition, and the impact of tokenization strategies in OCR for complex scripts.

Optical Character Recognition. Text recognition has evolved from traditional connectionist temporal classification (CTC)−based models to more sophisticated sequence-to-sequence (seq2seq) approaches. Early CTC frameworks, such as CRNN (Shi et al., 2017), combined CNN-based visual feature extraction with recurrent networks for temporal modeling. Subsequent methods expanded on this by incorporating attention modules, alternative visual encoders, and advanced normalization techniques to address irregular text shapes and challenging image conditions (Gao et al., 2019; Shi et al., 2018).

The introduction of transformer architectures (Vaswani et al., 2017) led to significant advances in seq2seq modeling. Transformers, with their self-attention mechanisms and positional encodings, handle variations in character scale, spatial arrangements, and bidirectional decoding more effectively (Zhang et al., 2020; Lee et al., 2020).

A more recent direction involves integrating language models (LMs) into recognition pipelines to enhance contextual understanding. TrOCR (Li et al., 2023), for example, leverages pretrained LMs as decoders using masked language modeling (MLM) objectives, while models like MaskOCR (Lyu et al., 2022) and ABINet (Fang et al., 2021) explore more sophisticated strategies for linguistic pre-training. DTrOCR (Fujitake, 2024) advances this line of work by omitting the encoder entirely and relying on an autoregressively pretrained decoder (GPT-2) (Radford et al., 2019).

In the domain of handwritten text recognition (HTR), these trends are paralleled by augmenting traditional RNN+CTC architectures with attention and language modeling components (Memon et al., 2020; Michael et al., 2019).

Bengali Handwritten Text Recognition. Most existing research on Bengali handwritten text recognition has focused on constrained tasks, such as recognizing isolated characters or digits (Sufian et al., 2022). A few works have extended this to word-level recognition using CRNN-based models trained with the CTC loss (Hossain et al., 2022; Basher et al., 2023). However, despite the growing prominence of transformer-based architectures in other languages, such methods remain largely unexplored for Bengali HTR.

Impact of Tokenization in OCR. Bengali script comprises approximately 13,000 graphemes (Alam et al., 2021)—substantially more than English, which has around 250. This makes the consideration of graphemes in tokenization a particularly important design decision in Bengali OCR. Basher et al. (2023) have shown empirically that grapheme-level tokenization yields improved performance in HTR compared to conventional character-level tokenization for Bengali.

3 Methodology

The system pipeline consists of two primary components: a text detection module, which segments full-page images into individual word images, and a text recognition module, which adopts a decoder-

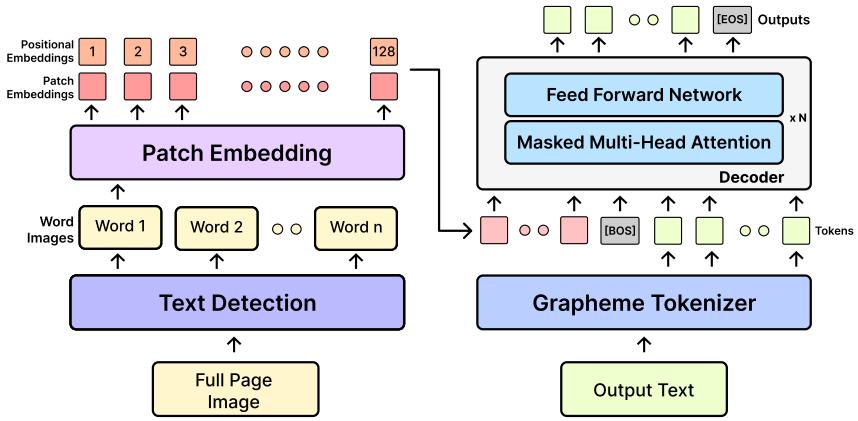


Figure 1: Architecture of the system pipeline. The pipeline consists of a text detection module and a text recognition module, which includes a patch embedding layer, a decoder-only transformer, and a grapheme-based tokenizer.

only architecture, beginning with a patch embedding layer followed by a decoder-only transformer. To enhance recognition accuracy for Bengali script, a grapheme-based tokenizer is integrated into the decoder. The overall architecture of the system is illustrated in Figure 1.

3.1 Text Detection

The task of recognizing handwritten text from full-page images typically requires large annotated datasets, which are often unavailable for Bengali. Consequently, most HTR methods operate at the line or word level. Given the constraints of our data setting, we adopt a word-level recognition approach.

The system begins by applying a text detection module to segment the full-page image into individual word images. For this, we employ BN-DRISHTI (Jubaer et al., 2023), a publicly available text detection model based on the YOLO framework (Redmon et al., 2016).

BN-DRISHTI follows a two-step line segmentation process. In the first step, line regions are identified directly from the full-page image. However, due to the curvilinear nature of Bengali handwriting, these preliminary detections often include spurious boundaries or lines above or below the actual lines. To correct for this, Hough line and Affine transformations are applied to normalize skew and straighten the text baselines. Preprocessing operations such as binarization and morphological filtering are then employed to suppress noise and refine the line boundaries. In the second step, the refined line images are re-segmented, yielding more accurate line-level outputs. Finally, individual word images are segmented from the refined text lines and forwarded to the recognition module.[3]

3.2 Text Recognition

We adopt a decoder-only text recognition module, in contrast to the encoder-decoder architecture commonly used in OCR. This module builds upon the open-source implementation of DTrOCR[4] (Fujitake, 2024), which we adapt using a grapheme-based tokenizer tailored for Bengali handwriting.

3.2.1 Patch Embedding

Transformers cannot directly process raw image inputs. Following the approach of ViT (Dosovitskiy et al., 2021), each input image of size $(3, H_0, W_0)$ is first resized to $(3, H, W)$ and then divided into non-overlapping patches of size (p_h, p_w), resulting in $N = (H/p_h) \times (W/p_w)$ patches. Each patch is linearly projected into a D-dimensional embedding. Positional encodings $P \in \mathbb{R}^{N \times D}$ are then added to retain spatial information. The resulting sequence is passed to the decoder module for autoregressive token prediction.

3.2.2 Decoder

The decoder module generates text directly from the embedded image sequence, bypassing the encoder used in traditional encoder–decoder architectures to extract visual features from raw images. This design eliminates the need for cross-attention and reduces both model parameters and computational cost. Our decoder module uses the architecture of Generative Pretrained Transformers (GPT) (Radford et al., 2018, 2019) to recognize

[3] Since we employ the exact model without any modification, we refer the interested readers to the BN-DRISHTI paper (Jubaer et al., 2023) for detailed methodology and evaluation results of the detection model.

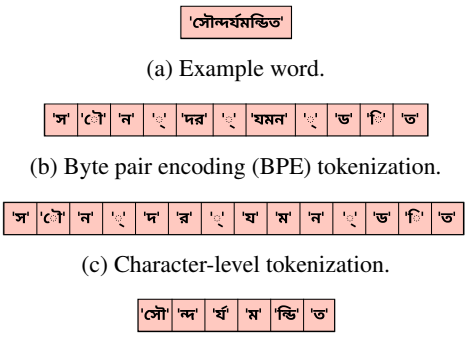
[4] https://github.com/arvindrajan92/DTrOCR

handwritten text. Specifically, we adopt a decoder-only architecture that consists solely of standard Transformer decoder blocks (Vaswani et al., 2017). In contrast to models such as TrOCR (Li et al., 2023) and DTrOCR (Fujitake, 2024), we do not rely on pretrained Bengali language models. Instead, our decoder is trained from scratch, making it suitable for low-resource settings.

After the embedded image sequence, a `[BOS]` token is appended and the resulting sequence is passed into the decoder. Tokens are then generated autoregressively until a `[EOS]` token is produced. The final output from the decoder is projected onto the vocabulary space and passed through a softmax layer for token prediction.

3.2.3 Tokenizer

A *grapheme* is the smallest visually distinguishable unit in a writing system. Transformer-based OCR models (Li et al., 2023; Fujitake, 2024) typically rely on default tokenizers such as Byte Pair Encoding (BPE) (Sennrich et al., 2016), which segments text into subwords or characters based on frequency statistics (see Figure 2b). These subword units often span multiple characters and are visually unstable in handwritten text: even slight variations in a single character can distort the entire token.



(a) Example word.

(b) Byte pair encoding (BPE) tokenization.

(c) Character-level tokenization.

(d) Grapheme-based tokenization.

Figure 2: Comparison of tokenization strategies for Bengali HTR using the example word shown in (a).

This challenge is amplified in Bengali, where writing styles vary widely and many graphemes are rendered with ligatures or conjuncts. Subword or character-level tokenization becomes unreliable without access to large, diverse training data. Furthermore, the visual order of Bengali text often diverges from its logical sequence, complicating alignment (see Figure 2c). Prior work (Basher et al., 2023) has shown that character-level tokenization fails to capture these structures effectively.

To address these challenges, we adopt a grapheme-based tokenizer that enforces a one-to-one mapping between visually coherent units and model output tokens (see Figure 2d). Specifically, we integrate BnGraphemizer (Basher et al., 2023), a trie-based grapheme tokenizer designed for Bengali, into our decoder-only transformer pipeline.

3.3 Training

The training process consists of two stages: pre-training on large-scale synthetic data and fine-tuning on real-world, human-annotated samples.

First, we employ a synthetic data generation tool[5] adapted from the VRD framework (Lauar and Laurent, 2024) to construct diverse datasets (as detailed in Section 4.1) for pre-training. Pre-training itself is conducted in two phases, first on line-level images and then on word-level images. In the initial phase, the training on synthetic images containing single lines of handwritten text helps the model learn high-level structure. In the latter phase, the model is further trained on word-level images. Line-level training is particularly useful in early stages, as text detection models may group multiple words into a single detection region, especially in cursive or compact writing styles.

Second, we fine-tune the pretrained model on human-annotated Bengali handwriting datasets (as detailed in Section 4.1), as previous works (Baek et al., 2021; Bautista and Atienza, 2022) have shown that synthetic-only dataset is generally insufficient for generalization to real-world inputs—an issue heightened in Bengali's complex script.

4 Experiments

In this section, we detail the experimental setup, describe the evaluation metrics, and present the results of our study.

4.1 Experimental Setup

We outline the experimental setup here, covering the details of training data and evaluation benchmarks, training configurations, and implementation details used in both the pre-training and fine-tuning stages.

4.1.1 Training Data and Benchmarks

For pre-training, we generate synthetic datasets comprising 4.5 million line-level samples and 7

[5] https://github.com/tahsinchoudhury/GlyphScribe

Model	Tokenizer	Pretrained LLM	#Parameters	BN-HTRd		Bongabdo	
				CER(%)	WER(%)	CER(%)	WER(%)
Line_BPE_Pretrained	BPE	✓	162M	38.34	51.83	65.04	77.65
Line_BPE_Random	BPE	✗	162M	26.17	39.22	46.91	63.85
Line_BnG_Random	BnGraphemizer	✗	87M	29.58	42.89	50.69	68.02
Word_BPE_Pretrained	BPE	✓	162M	7.57	16.30	11.61	26.99
Word_BPE_Random	BPE	✗	162M	6.68	15.05	10.09	25.39
Word_BnG_Random	BnGraphemizer	✗	87M	**6.19**	**14.20**	**8.68**	**23.56**

Table 1: Model configuration details and recognition performance (CER, WER) on the BN-HTRd and Bongabdo datasets. Model names follow the format *[Input granularity]_[Tokenizer]_[Decoder initialization]*. Here, *Line* level models take entire handwritten line images as input and predict the full line text, while *Word* level models take cropped word images as input and predict the word text. *BPE* and *BnG* denote the Byte-Pair Encoding and BnGraphemizer tokenizers, respectively. *Pretrained*/*Random* indicates whether the decoder was initialized from a pretrained Bengali language model or with random weights. Among all variants, **Word_BnG_Random** (our proposed word-level model with grapheme tokenizer, trained from scratch) achieves the lowest CER and WER.

million word-level samples. The text content is sourced from Bengali Wikipedia, the Bangla-NMT dataset (Hasan et al., 2020), and a publicly available Bengali dictionary (Kamal, 2018). These sources provide a wide range of vocabulary and writing styles. To enhance realism, our synthetic data generator introduces artifacts such as handwritten-style fonts, wavy or bent lines, Gaussian blur, and partial character fragments that mimic cropping effects. Representative examples are illustrated in Appendix A.

To fine-tune and evaluate our model, we use six human-annotated datasets: BN-HTRd (Rahman et al., 2023; Jubaer et al., 2023), BanglaWriting (Mridha et al., 2021), BanglaLekha-Isolated (Biswas et al., 2017), IIIT-Indic (Jindal et al., 2021), ICDAR 2024 (IIIT Hyderabad and CVIT Lab, 2024), and Bongabdo (Islam et al., 2023).

Dataset	# Words
BN-HTRd	104,854
BanglaWriting	21,234
BanglaLekha-Isolated (Digits only)	19,748
IIIT-Indic	113,075
ICDAR 2024	79,663
Total	**338,574**

Table 2: Summary of human-labeled datasets used to fine-tune the word-level models.

We consider two model variants: line-level models, which take entire handwritten line images as input and predict the full line text, and word-level models, which take cropped word images as input and predict the text for that word. For fine-tuning, the line models use 13,912 real handwritten line images from the BN-HTRd dataset (Rahman et al., 2023). In contrast, the word models are fine-tuned on a larger collection of approximately 340,000 (see Table 2 for details) real handwritten word images aggregated from BN-HTRd (Rahman et al., 2023), BanglaWriting, BanglaLekha-Isolated, IIIT-Indic, and ICDAR 2024. For evaluation, we benchmark on 805 full-page images from the "automatic annotation" split of the extended BN-HTRd dataset (Jubaer et al., 2023) and 111 images from the Bongabdo dataset.

4.1.2 Implementation Details

We implement our decoder using the GPT-2 architecture from HuggingFace (Wolf et al., 2020), consisting of 12 transformer layers with a hidden size of 768 and 12 self-attention heads. The maximum sequence length is set to 256 tokens. To better suit the properties of Bengali handwritten text, we replace the default byte-pair encoding tokenizer with the BnGraphemizer tokenizer.

Each input image is resized to a fixed resolution of 32×128 pixels and divided into non-overlapping patches of size 4×8, yielding 128 image tokens per word. The model is pretrained using a batch size of 32 with the Adam optimizer and a learning rate of 1×10^{-4}. During fine-tuning, the same batch size and optimizer are used, but the learning rate is reduced to 5×10^{-6}.

All training and inference experiments are conducted using PyTorch on a single NVIDIA GeForce RTX 3080 GPU (see Appendix C for more details).

4.2 Evaluation Metrics

We evaluate our handwriting recognition system using two widely adopted metrics for handwritten text

System	BN-HTRd		Bongabdo	
	CER (%)	WER (%)	CER (%)	WER (%)
Gemini 2.5 Flash[6]	19.39	32.49	37.42	56.39
Imagetotext.info[7]	20.90	37.10	14.36	31.14
Ours	**6.19**	**14.20**	**8.68**	**23.56**

Table 3: Comparison of cloud-based OCR systems and our proposed model on Bengali HTR.

Filtered	BN-HTRd		Bongabdo	
	CER (%)	WER (%)	CER (%)	WER (%)
No	6.19	14.20	8.68	23.56
Yes	**4.58**	**12.59**	**8.02**	**22.95**

Table 4: Comparison of recognition performance with and without filtering images where the text detection module failed to segment words accurately.

recognition (HTR): Character Error Rate (CER) and Word Error Rate (WER). CER measures the proportion of character-level errors—substitutions, deletions, and insertions—required to transform the predicted sequence into the ground truth, while WER computes the same at the word level. Formal definitions of both metrics are provided in Appendix B.

4.3 Experimental Results

We now discuss the results of our model across ablations, comparisons with existing OCR systems and prior HTR models, an analysis of its sensitivity to text detection quality and inference speed.

Model Configuration Analysis. We compare multiple model configurations to evaluate the effects of input granularity, decoder initialization, and tokenization. Specifically, we study line-level vs. word-level recognition, pretrained Bengali language model vs. randomly initialized decoders, and subword-level Byte Pair Encoding (BPE) vs. the grapheme-level BnGraphemizer tokenizer. While BPE is used with the pretrained LLM, BnGraphemizer is evaluated only with randomly initialized decoders, as no pretrained Bengali LLM exists for this tokenizer.

The line-level models are pretrained on synthetic line images, while word-level models undergo additional pretraining on synthetic word images. Line-level models are fine-tuned on handwritten line images, while word-level models are fine-tuned on a larger set of handwritten word images, as discussed in Section 4.1.

As shown in Table 1, word-level models outperform line-level ones, which can be attributed to the abundance of word-level training data and the more nuanced nature of line-level data. Our proposed model—Word_BnG_Random—achieves the lowest CER and WER without using a pretrained Bengali LLM. The grapheme-based tokenizer further improves accuracy while reducing model size from 162M to 87M, demonstrating both effectiveness and parameter efficiency.

Comparison with Existing OCR Systems. We compare our proposed model with two existing OCR systems: Gemini 2.5 Flash and Imagetotext.info. As shown in Table 3, our model achieves the lowest CER and WER, outperforming both systems by a large margin. Gemini 2.5 Flash is selected as a representative vision-language model (see Appendix D for prompt details) due to its strong performance on Bengali handwritten text, substantially surpassing models such as ChatGPT and Claude on this task. We exclude other Google Vision API models, as Gemini consistently outperforms them. Imagetotext.info is included for its support for Bengali handwriting and its accessible API, which enables automated benchmarking. Other OCR services were excluded based on lack of Bengali handwriting support, absence of APIs, or poor performance in preliminary evaluations.

Dependency on Text Detection Model. To assess the sensitivity of our system to the quality of word segmentation, we manually inspected the BN-HTRd and Bongabdo datasets and identified a small number of samples where the text detection model failed to accurately segment words. After filtering out these cases, our text recognition model demonstrated improved performance, as shown in Table 4. These findings suggest that the end-to-end pipeline is affected by errors in text detection, and that further gains in recognition accuracy may be achieved by incorporating a more precise word segmentation module.

Comparison with Prior Models. Table 5 presents cross-dataset evaluation results comparing our proposed model with prior Bengali HTR approaches, including LILA-BOTI (Hossain et al., 2022), CRNN-CT, and CRNN-BnG (Basher et al., 2023). We adopt cross-dataset evaluation to ensure consistency with prior work, which report performance under the same setting. This approach

[6]Although Gemini 2.5 Pro yields slightly better results, we use Flash due to strict API limits in the Pro free tier.

[7]https://www.imagetotext.info

Datasets	Model	CER (%)	WER (%)
BW (train) BN-HTRd (test)	LILA-BOTI*	25.92	51.62
	CRNN-CT*	19.94	44.10
	CRNN-BnG*	30.53	48.41
	Ours	**18.04**	**33.20**
BN-HTRd (train) BW (test)	LILA-BOTI*	15.54	36.78
	CRNN-CT*	**11.12**	29.43
	CRNN-BnG*	15.80	29.18
	Ours	12.66	**24.79**

Table 5: Cross-dataset evaluation on BW (BanglaWriting) and BN-HTRd datasets. The asterisk (*) indicates results reported from (Basher et al., 2023).

provides a fair basis for comparison and reflects the models' ability to generalize across different handwriting corpora. When fine-tuned on BW (BanglaWriting) (Mridha et al., 2021) and evaluated on BN-HTRd (Rahman et al., 2023), our model achieves the lowest CER and WER. In the reverse setting, it obtains the best WER, while the CER is slightly higher.

Inference Speed. We conducted experiments on the Bongabdo dataset to compare the inference speeds of different models, as reported in Table 6. Each experiment was repeated five times to ensure statistical robustness, and the average inference speed was reported as the final result. This set of experiments was performed on the aforementioned devices, which were also used for model training and other experiments. As expected, our proposed model with 87M parameters achieves a higher inference speed than the 162M parameter variant. This demonstrates the practicality of our model for deployment in low-resource computational settings while delivering faster results to end users.

Model	#Params	#Words	Time	Speed
Word_BPE_Random	162M	17,217	2074.32s	8.30 words/s
Word_BnG_Random	87M	17,217	**1729.59s**	**9.95** words/s

Table 6: Inference time on the Bongabdo dataset.

5 System Description

We deploy our Bengali handwriting recognition system as a Flask-based REST API, enabling external access through a lightweight and modular interface. The backend supports asynchronous task execution using Celery, with Redis serving as the message broker. To ensure reproducibility and ease of deployment, the entire system is containerized with Docker, and all components are orchestrated via Docker Compose. The inference engine runs on an NVIDIA GeForce RTX 3080 GPU.

The system supports a variety of input formats, including individual image files (PNG, JPG, JPEG) as well as multi-page PDF documents. Once uploaded, documents are processed through the detection and recognition pipeline described in Sections 3.1 and 3.2, respectively, where full-page handwritten images are segmented into word-level regions and transcribed using a decoder-only transformer model. For each image or PDF page, the recognized text is displayed in an editable text area, allowing users to make corrections as needed. In the case of PDFs, the system provides per-page navigation to support multi-page review. Users can export the extracted text in either plain text (`.txt`) or Microsoft Word (`.docx`) format.

The web interface features a document upload panel on the left and a text output display area on the right. Snapshots of the user interface, along with representative use cases, are provided in Appendix E.

6 Conclusions

We present a resource-efficient Bengali handwritten text recognition system that operates at the word level without relying on a pretrained large language model as the decoder. Our approach leverages a decoder-only transformer combined with a grapheme-based tokenizer tailored for Bangla script. Experimental results demonstrate that our method outperforms existing OCR systems and prior Bengali HTR models, achieving state-of-the-art accuracy despite being trained on limited annotated data.

7 Limitations and Future Work

Our system has two main limitations: its training data mainly consists of handwritten text on white backgrounds, limiting adaptability to more varied real-world scenarios; and the scarcity of large-scale pretrained GPT-2 models for Bengali limits the breadth of the experiments. Future work can include expanding the synthetic dataset to cover diverse backgrounds and noise artifacts, pre-training auto-regressive language models on extensive Bengali text, experimenting with other auto-regressive language models (such as Llama), and refining text detection to reduce segmentation errors. To support future research in Bengali HTR, we publicly release our complete pipeline under the MIT license.

Ethical Considerations

Our system does not retain any user-submitted documents or images. All uploads are processed temporarily and permanently deleted after a short duration to ensure user privacy. The datasets used for training and evaluation are publicly available and intended for research purposes. No personal or sensitive content was included in the model development process.

We promote responsible use of the system in archival, educational, and administrative contexts. Its intended applications include the digitization of historical or administrative documents, support for literacy development, and automation of handwritten workflows in institutional settings. It is not designed for surveillance, profiling, or other harmful purposes.

Acknowledgements

The authors acknowledge the support provided by the Bangladesh Bureau of Educational Information and Statistics (BANBEIS) and the University of Dhaka during the course of this research.

References

S. Alam, T. Reasat, A. S. Sushmit, S. M. Siddique, F. Rahman, M. Hasan, and A. I. Humayun. 2021. A large multi-target dataset of common bengali handwritten graphemes. In *Document Analysis and Recognition – ICDAR 2021*, pages 383–398. Springer International Publishing.

M. A. Azad, H. S. Singha, and M. M. H. Nahid. 2020. Bangla handwritten character recognition using deep convolutional autoencoder neural network. In *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, pages 295–300.

Jeonghun Baek, Yusuke Matsui, and Kiyoharu Aizawa. 2021. What if we only use real datasets for scene text recognition? toward scene text recognition with fewer labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3113–3122.

Mohammad Jahid Ibna Basher, Mohammad Raghib Noor, Sadia Afroze, and Ikbal Ahmed. 2023. Bngraphemizer: A grapheme-based tokenizer for bengali handwritten text recognition. In *2023 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering*, Sree Chitra Thirunal College of Engineering, Thiruvananthapuram, Kerala, INDIA.

Darwin Bautista and Rowel Atienza. 2022. Scene text recognition with permuted autoregressive sequence models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 178–196.

Mithun Biswas, Rafiqul Islam, Gautam Kumar Shom, Md. Shopon, Nabeel Mohammed, Sifat Momen, and Anowarul Abedin. 2017. Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten bangla isolated characters. *Data in Brief*, 12:103–107.

A. Chaudhury, P. S. Mukherjee, S. Das, C. Biswas, and U. Bhattacharya. 2022. A deep ocr for degraded bangla documents. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 21(5).

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Shancheng Fang, Hongtao Xie, Yuxin Wang, Zhendong Mao, and Yongdong Zhang. 2021. Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7098–7107.

Masato Fujitake. 2024. Dtrocr: Decoder-only transformer for optical character recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 8025–8035.

Y. Gao, Y. Chen, J. Wang, M. Tang, and H. Lu. 2019. Reading scene text with fully convolutional sequence modeling. *Neurocomputing*, 339:161–170.

A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural network. In *Proceedings of the 23rd international conference on Machine learning*.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M. Sohel Rahman, and Rifat Shahriyar. 2020. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2612–2623, Online. Association for Computational Linguistics.

M. I. Hossain, M. Rakib, S. Mollah, F. Rahman, and N. Mohammed. 2022. Lila-boti: Leveraging isolated letter accumulations by ordering teacher insights for bangla handwriting recognition. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1770–177.

M. T. Hossain, M. W. Hasan, and A. K. Das. 2021. Bangla handwritten word recognition system using convolutional neural network. In *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–8.

IIIT Hyderabad and CVIT Lab. 2024. Icdar 2024 handwritten document recognition dataset. https://ilocr.iiit.ac.in/icdar_2024_hwd/.

Md Majedul Islam, Avishek Das, Ibna Kowsar, AKM Shahariar Azad Rabby, Nazmul Hasan, and Fuad Rahman. 2023. Towards full-page offline bangla handwritten text recognition using image-to-sequence architecture. In *2023 IEEE International Conference on Big Data (Big Data)*, pages 1061–1067. IEEE.

Milan Jindal, Pratyush Kumar, and C.V. Jawahar. 2021. iiit-indic-hw-words: A dataset for indic handwritten text recognition. *arXiv preprint arXiv:2105.04020*.

Sheikh Mohammad Jubaer, Nazifa Tabassum, Md Ataur Rahman, and Mohammad Khairul Islam. 2023. Bn-drishti: Bangla document recognition through instance-level segmentation of handwritten text images. *arXiv preprint arXiv:2306.09351*.

M. Kamal. 2018. Minhaskamal/bengalidictionary: A large collection of bengali words. https://github.com/MinhasKamal/BengaliDictionary.

Filipe Lauar and Valentin Laurent. 2024. Spanish trocr: Leveraging transfer learning for language adaptation. *Preprint*, arXiv:2407.06950.

J. Lee, S. Park, J. Baek, S.J. Oh, S. Kim, and H. Lee. 2020. On recognizing texts of arbitrary shapes with 2d self-attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 546–547.

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2023. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13094–13102.

Hengyuan Lyu, Chengquan Zhang, Shanshan Liu, Meina Qiao, Yangliu Xu, Liang Wu, Kun Yao, Junyu Han, Errui Ding, and Jingdong Wang. 2022. Maskocr: Text recognition with masked encoder-decoder pretraining. *arXiv preprint arXiv:2206.00311*.

J. Memon, M. Sami, R. A. Khan, and M. Uddin. 2020. Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr). *IEEE Access*, 8:142642–142668.

J. Michael, R. Labahn, T. Grüning, and J. Zöllner. 2019. Evaluating sequence-to-sequence models for handwritten text recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1286–1293. IEEE.

Md Forhad Mridha, Abu Quwsar Ohi, Md Ameer Ali, Md I Emon, and Md Mohsin Kabir. 2021. Banglawriting: A multi-purpose offline bangla handwriting dataset. *Data in Brief*, 34:106633.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI*. Technical report.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI. Technical report.

Md Ataur Rahman, Nazifa Tabassum, Manash Paul, Rajib Pal, and Mohammad Khairul Islam. 2023. Bn-htrd: A benchmark dataset for document level offline bangla handwritten text recognition (htr) and line segmentation. In *Computer Vision and Image Analysis for Industry 4.0*, pages 1–16. CRC Press.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

F. B. Safir, A. Q. Ohi, M. F. Mridha, M. M. Monowar, and M. A. Hamid. 2021. End-to-end optical character recognition for bengali handwritten words. In *2021 National Computing Colleges Conference (NCCC)*, pages 1–7.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

F. Sheng, Z. Chen, and B. Xu. 2019. Nrtr: A no-recurrence sequence-to-sequence model for scene text recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 781–786. IEEE.

B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai. 2018. Aster: An attentional scene text recognizer with flexible rectification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2035–2048.

Baoguang Shi, Xiang Bai, and Cong Yao. 2017. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304.

A. Sufian, A. Ghosh, A. Naskar, F. Sultana, J. Sil, and M. H. Rahman. 2022. Bdnet: Bengali handwritten numeral digit recognition based on densely connected convolutional neural networks. *Journal of King Saud University - Computer and Information Sciences*, 34(6, Part A):2610–2620.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

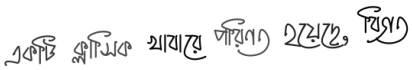
S. Wang, Y. Wang, X. Qin, Q. Zhao, and Z. Tang. 2019. Scene text recognition via gated cascade attention. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1018–1023. IEEE.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

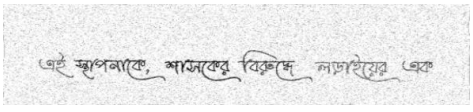
Jiaxing Zhang, Canjie Luo, Lianwen Jin, Tianwei Wang, Ziyan Li, and Weiying Zhou. 2020. Sahan: Scale-aware hierarchical attention network for scene text recognition. *Pattern Recognition Letters*, 136:205–211.

A Synthetic Images

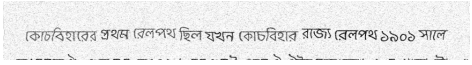
To make synthetic images resemble real handwritten samples, we introduce artifacts such as wavy and bent text, Gaussian blur, and partial character fragments simulating real-world cropping effects during data generation. Representative examples are illustrated in Figure 3.



(a) Bent line with a clear background.



(b) Line image with Gaussian blur applied.



(c) Wavy line with partial character fragments.

(d) Word with clear background.

Figure 3: Representative examples generated by the synthetic image generator.

B Definitions of the Evaluation Metrics

We provide the definitions of Character Error Rate (CER) and Word Error Rate (WER) below.

CER. CER measures the proportion of character-level errors—substitutions (S), deletions (D), and insertions (I)—required to transform the predicted text sequence (hypothesis) into the reference text (ground truth), normalized by the total number of characters in the reference (N). Formally,

$$\text{CER} = \frac{S + D + I}{N}$$

WER. WER measures the proportion of word-level errors—substitutions (S), deletions (D), and insertions (I)—needed to convert the predicted word sequence into the reference word sequence, divided by the total number of words in the reference (N). Formally,

$$\text{WER} = \frac{S + D + I}{N}$$

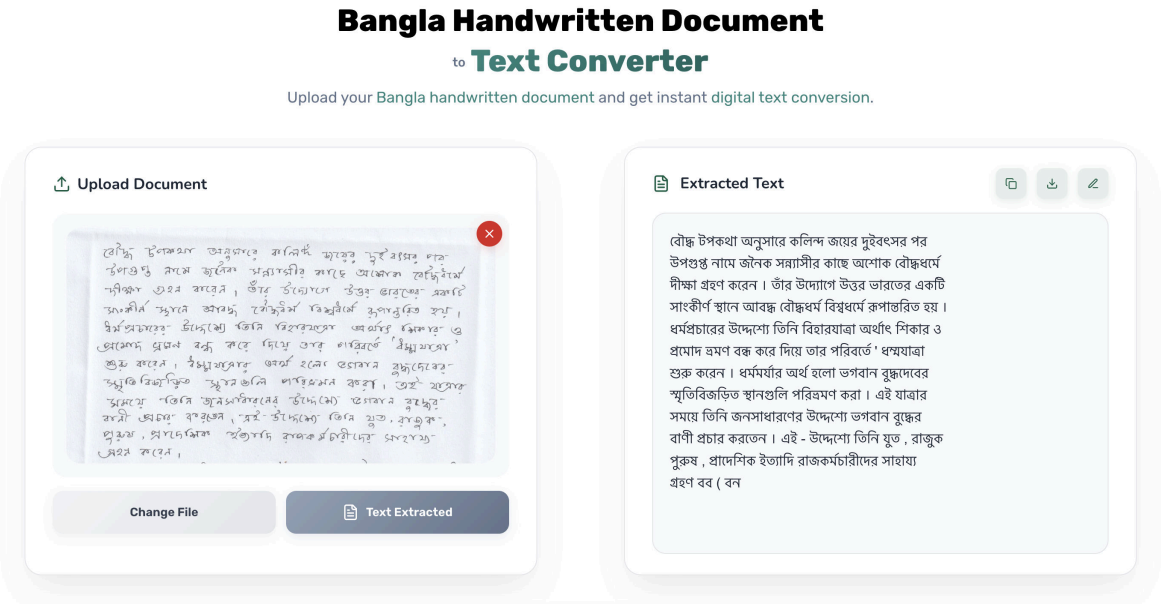
C Training Details

Our model was pretrained and fine-tuned on a single NVIDIA GeForce RTX 3080 GPU with a batch size of 32. The first-stage pretraining on 4.5M synthetic line-level images requires 13 hours over 2 epochs, while the second-stage pretraining on 7M synthetic word-level images takes 9 hours for a single epoch. The model is then fine-tuned on real handwritten datasets for 4 epochs, completing in less than 2 hours.

D LLM Evaluation

The prompt we use to transcribe an image using an external LLM is provided below.

> You are an optical character recognition (OCR) tool. Your task is to transcribe the handwritten Bengali text shown in the provided image directly and exactly, without making any corrections, translations, or modifications. Extract the text precisely as it appears, preserving original spelling, grammar, and formatting.
> Output only the text found in the image, with no additional comments, explanations, or formatting.



(a) Example of the interface after uploading an image.

(b) Example of the interface after uploading a PDF.

Figure 4: Screenshot of the Bangla Handwritten Document to Text Converter web interface, showing the document upload area on the left and the extracted text display area on the right.

E Illustrative Outputs from the System

Figure 4 depicts the system's user interface along with sample use-cases. A video demonstrating how to use the system is available on YouTube[8].

[8] https://youtu.be/ckgWBHQarxc