Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Telugu Handwritten Word Recognition Using an Autoregressive Transformer Decoder with Joint CTC-Attention Training

Sakilam Bhargav

23211a67a7@bvrit.ac.in

B.V. Raju Institute of Technology

Thatha Nikitha

B.V. Raju Institute of Technology

Sabavat Vinod Nayak

B.V. Raju Institute of Technology

R.Venkata Ramana Chary

B.V. Raju Institute of Technology

Research Article

Keywords:

Posted Date: November 10th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10868282/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Telugu Handwritten Word Recognition Using an Autoregressive Transformer Decoder with Joint CTC-Attention Training

Sakilam Bhargav[1], Thatha Nikitha[1], Sabavat Vinod Nayak[1], R. Venkata Ramana Chary[1*]

[1]Dept. of CSE (Data Science), B.V. Raju Institute of Technology, Narsapur, Telangana, India.

*Corresponding author(s). E-mail(s): ramanachary.rv@bvrit.ac.in;
Contributing authors: 23211a67a7@bvrit.ac.in; 23211a67b6@bvrit.ac.in; 23211a67a6@bvrit.ac.in;

Abstract

Telugu handwriting recognition remains difficult, largely because of the script's syllabic structure. Consonants fuse with vowel diacritics and stack into conjuncts via the virama; the resulting compound glyphs bear little visual resemblance to their constituent characters. We compare a CNN-BiLSTM-CTC baseline with an autoregressive Transformer decoder trained under a joint CTC-attention loss on the IIIT-HW-Telugu benchmark (80,693 training images, 17,910 test images). Both decoders share a stride-modified ResNet-18 encoder that preserves horizontal resolution for sequential decoding. We also construct a character bigram transition matrix from the training labels and apply it as a soft constraint during inference. The best Transformer configuration achieves 3.66% character error rate (CER), a 6.4% relative reduction over the CTC baseline's 3.91%. Two details proved critical for convergence: scaling token embeddings by $\sqrt{d_{\text{model}}}$ and tying input–output embedding weights. An earlier decoder lacking both still trained but plateaued at 4.89% CER—below the CTC baseline—though that comparison confounds several hyperparameters and is not an isolated ablation. Error analysis reveals a practical trade-off: CTC scatters small substitutions across many samples, while the Transformer is more accurate on average but occasionally exhibits what we term *autoregressive collapse*, confidently emitting a valid Telugu word unrelated to the input image. Code and trained weights are publicly available.

Keywords: Handwritten text recognition, Telugu, CTC, Transformer, Autoregressive decoding, Indic scripts, Linguistic constraints

1 Introduction

Handwritten text recognition (HTR) for Latin scripts now routinely achieves single-digit character error rates on standard benchmarks such as IAM [1] and RIMES [4]. Indic scripts lag behind. Telugu, spoken natively by over 80 million people and designated one of India's classical languages [17], has received comparatively little HTR attention relative to its speaker population. As the most widely spoken Dravidian language, used extensively in government and education across Andhra Pradesh and Telangana, Telugu represents a significant gap in HTR coverage.

The difficulty is largely structural. Telugu has 36 base consonants (*hallulu*), 16 vowels (*acchulu*), and vowel diacritics (*matras*) that attach to consonant glyphs and alter their shape. The virama character (*halant*, U+0C4D) stacks consonants into conjuncts (*vattu* forms), producing compound glyphs that can be visually unrelated to their constituent consonants. A single word may contain multiple conjuncts, each combining two or three base consonants into a glyph whose visual form must be learned independently. Latin letters, by contrast, retain roughly the same shape regardless of context. Wide variation in individual handwriting styles compounds the challenge.

Current HTR systems generally follow one of two decoding strategies. CTC [2] avoids character-level bounding boxes by having the encoder emit a distribution at every time step; dynamic programming then marginalises over all alignments consistent with the target label sequence. CTC is fast and effective for left-to-right scripts, but each output position is predicted independently, so the decoder cannot model dependencies between adjacent characters. Autoregressive (AR) Transformer decoders [1] generate characters one at a time, conditioning each prediction on the full history of previously emitted tokens. This enables the decoder to capture inter-character dependencies—for example, that a particular vowel diacritic rarely follows a certain consonant cluster in Telugu—at the cost of slower inference and exposure bias, where an early decoding error propagates into all subsequent predictions.

With sufficient pre-training data, Transformers outperform CTC on English HTR. TrOCR [4] and PARSeq [5] demonstrated this on benchmarks comprising millions of images. For low-resource scripts the evidence is less clear: Gongidi and Jawahar [10] observed that CTC remains competitive on Indic languages when labelled data is scarce. The ICDAR 2023 competition on handwritten Indic text recognition [19] further highlighted that performance gaps between CTC-based and attention-based systems narrow considerably when training data is limited to tens of thousands of samples. This motivates the central question of our work: *given roughly 80k Telugu word images, does a Transformer decoder outperform a well-tuned CTC baseline, and by how much?*

A secondary question concerns the role of linguistic knowledge. Telugu's phonotactic rules constrain which characters can follow one another—for instance, certain vowel diacritics never appear after specific consonant clusters. We investigate whether a character-level bigram transition matrix, derived from training-set statistics, can improve decoding accuracy when applied as a soft constraint at inference time.

Specifically, we make the following contributions:

1. A shared ResNet-18 visual encoder with modified stride patterns that preserves horizontal resolution for both the CTC and Transformer decoders, enabling a controlled comparison between the two decoding strategies.
2. An autoregressive Transformer decoder trained with a joint CTC-attention loss, along with two architectural details—embedding scaling and weight tying—that in our experience separated a decoder that merely trains from one that outperforms the CTC baseline.
3. A data-driven Telugu character transition matrix built from training-set bigram counts, tested as a soft decoding constraint.
4. To our knowledge, the lowest lexicon-free CER reported on the IIIT-HW-Telugu benchmark [10]: 3.66%, down from the 4.58% obtained by Dutta et al. [9].
5. A methodological caution for Indic HTR: because conditioning on the presence of a virama also conditions on word length, CER computed over compound and simple subsets is not directly comparable, and the apparent advantage on compound words disappears once measured at the word level.

The remainder of this paper is organised as follows. Section 2 reviews related work on CTC-based and attention-based HTR, with a focus on Indic scripts. Section 3 describes the proposed architecture. Section 4 details the experimental setup. Section 5 presents and analyses results. Section 6 discusses implications and limitations, and Section 7 concludes.

2 Related Work

2.1 CTC-Based Handwriting Recognition

CTC [2] remains the dominant loss for HTR because it requires only image–transcript pairs, not character-level bounding boxes. The CRNN pipeline of Shi et al. [6]—convolutional features fed into a BiLSTM and decoded with CTC—established the standard baseline architecture and has been widely adopted across multiple scripts and languages. Puigcerver [7] later demonstrated that width-preserving convolutions improve the encoder by retaining more spatial information along the horizontal axis, and Scheidl et al. [8] integrated language-model-aware beam search into CTC decoding, showing that external knowledge can partially compensate for CTC's conditional independence assumption. Individually these refinements yield modest gains, but collectively they have kept CTC competitive with attention-based alternatives.

CTC's main limitation for Indic scripts is the conditional independence assumption: each output character depends only on the encoder at the corresponding time step. In Telugu, vowel diacritics are highly predictable given the preceding consonant, and CTC cannot exploit this regularity.

2.2 Attention and Transformer-Based HTR

Transformer architectures entered HTR via scene-text recognition. TrOCR [4] combines a pre-trained ViT encoder with a GPT-2-style decoder and reports strong results on several English benchmarks; its performance, however, depends on large-scale pre-training corpora that do not exist for most Indic scripts. PARSeq [5] trains a single

model on every permutation of character order, enabling both left-to-right and parallel decoding, which addresses the exposure-bias problem inherent in standard autoregressive models. To our knowledge, neither system has been evaluated on Telugu, so whether their gains transfer to a syllabic script remains open.

Attention-based encoder–decoder models have also been applied to HTR outside the Transformer framework. The attention mechanism lets the decoder select relevant spatial regions of the encoder output at each step, rather than relying on CTC's fixed left-to-right alignment. For Indic scripts, where diacritics appear above, below, or beside the base consonant, this spatial flexibility should in principle help the decoder locate each component independently. In practice, attention-based decoders are susceptible to alignment failures: the attention weights sometimes fail to form a monotonic pattern and instead attend to irrelevant positions, producing garbled output. The joint CTC-attention formulation (Section 2.4) was proposed specifically to mitigate this problem.

2.3 Indic Script Recognition

Published work on Telugu HTR specifically is limited. Dutta et al. [9] trained CNN-RNN-CTC pipelines on IIIT-HW-Telugu and reported 9.15% CER, reduced to 4.58% with IAM pre-training—a cross-script transfer result that suggests the CNN encoder learns general stroke features transferable across writing systems. Gongidi and Jawahar [10] introduced the broader IIIT-INDIC-HW-WORDS benchmark covering ten Indic scripts and obtained 6.41% CER on the Telugu subset with a CRNN; lexicon-constrained decoding lowered this to 1.52%, though such decoding is limited to a closed vocabulary and cannot recognise out-of-vocabulary words. The PLATTER system [11] addressed page-level Indic HTR but evaluated only on Hindi and Bangla, not Telugu.

The ICDAR 2023 competition on handwritten Indic text recognition [19] provided a standardised evaluation framework across multiple Indic scripts, including Telugu. Competition results showed that CTC-based systems remained competitive with attention-based systems when training data was limited, consistent with the observations of Gongidi and Jawahar. More recently, studies have explored synthetic data generation using GANs to augment limited handwritten Indic datasets [20], though the effectiveness of such approaches for Telugu specifically remains underexplored.

2.4 Joint CTC-Attention Training

Watanabe et al. [12] showed in speech recognition that adding an auxiliary CTC loss alongside an attention decoder accelerates convergence. The CTC branch encourages the encoder to produce roughly time-aligned features, giving the attention mechanism a simpler alignment to learn. Without this prior, the attention mechanism must discover monotonic left-to-right alignment entirely from the training signal, which can be slow and unstable for long sequences.

Michael et al. [13] applied the same joint-loss formulation to HTR and reported faster convergence and fewer attention-collapse episodes relative to pure attention training. They observed that the CTC branch acts as a regulariser, preventing the

encoder from learning representations incompatible with monotonic alignment. We adopt this formulation for our Transformer decoder.

3 Proposed Method

3.1 Problem Formulation

Given a handwritten word image $\mathbf{X} \in \mathbb{R}^{H \times W}$ (grayscale, $H = 64$, $W = 512$), the task is to predict the corresponding character sequence $\mathbf{y} = (y_1, y_2, \ldots, y_L)$ where each y_i belongs to a vocabulary $\mathcal{V}$ of 91 tokens (87 Telugu characters plus 4 special tokens: PAD, SOS, EOS, UNK). All ground-truth labels are NFC-normalised to ensure consistent Unicode representation of composed characters.

The recognition pipeline is modelled as:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{X}; \theta) \tag{1}$$

where θ denotes the model parameters. We decompose this into an encoder f_{enc} that maps the image to a feature sequence $\mathbf{h} = (h_1, \ldots, h_T)$ and a decoder f_{dec} that produces the output character sequence from $\mathbf{h}$.

3.2 System Overview

The system comprises four components, illustrated in Fig. 1: a shared CNN feature encoder, a CTC baseline decoder, an autoregressive Transformer decoder with an auxiliary CTC head, and an optional Telugu character-transition mask applied during inference.

3.3 CNN Feature Encoder

The visual encoder is a ResNet-18 [3] pre-trained on ImageNet, with modified spatial downsampling. An unmodified ResNet-18 reduces both height and width by a factor of 32; for a 512-pixel-wide input this would leave only 16 horizontal positions, insufficient for CTC or attention decoding of a typical 8–10 character Telugu word. We address this by setting the stride of the first convolution in layers 3 and 4 from $(2, 2)$ to $(2, 1)$:

$$\text{layer3[0].stride} : (2, 2) \rightarrow (2, 1) \tag{2}$$

$$\text{layer4[0].stride} : (2, 2) \rightarrow (2, 1) \tag{3}$$

Height is still reduced, but horizontal resolution is preserved, yielding feature maps of shape $[B, 512, 4, W/8]$. The input is grayscale, replicated to three channels to match the pre-trained weights. A 1×1 convolution projects the 512 channels to d_{model}, adaptive average pooling collapses the height dimension, and the result is a sequence of $T = W/8 = 64$ feature vectors. Sinusoidal positional encodings [1] are added to preserve ordering information.

The choice of ResNet-18 over deeper architectures (e.g., ResNet-50) was motivated by the relatively small size of our training set: deeper encoders showed

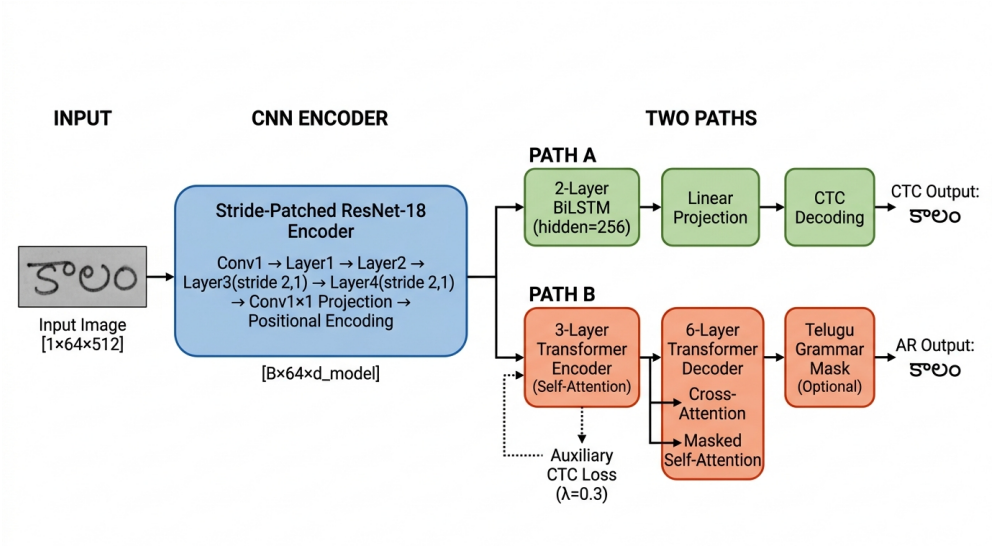


Fig. 1 System overview. The stride-patched ResNet-18 encoder turns a word image into a 64-step feature sequence, keeping the layer 3–4 strides at $(2, 1)$ rather than $(2, 2)$. Two decoder paths share this representation: **Path A** runs it through a 2-layer BiLSTM for CTC decoding, while **Path B** refines it with a 3-layer Transformer encoder, branches off an auxiliary CTC head (λ=0.3), and decodes autoregressively with a 6-layer Transformer decoder. An optional Telugu grammar mask penalises unlikely character transitions at inference time.

signs of overfitting in preliminary experiments without providing measurable CER improvement.

3.4 CTC Baseline Architecture

For the CTC baseline, we stack a 2-layer bidirectional LSTM (hidden size 256, giving 512-dimensional concatenated states) on top of the shared encoder features, then project those states with a linear head to logits over the 91-token vocabulary. Training uses CTC loss [2], with the PAD token (index 0) serving as the CTC blank symbol. BiLSTM weights are initialised with the orthogonal scheme. At test time, decoding is greedy: repeated tokens and blanks are collapsed in a single left-to-right pass.

The BiLSTM adds temporal context on top of the CNN features. Because the CNN encoder produces locally computed features even with the modified stride, dependencies spanning multiple characters are not captured until the BiLSTM propagates information across time steps.

3.5 Autoregressive Transformer Decoder

The Transformer decoder follows Vaswani et al. [1] with several modifications. Each of the $N = 6$ decoder layers contains masked self-attention, cross-attention over the encoder memory, and a feed-forward block with GELU activations. Layer normalisation is applied *before* each sub-layer (Pre-LN), as recommended by Xiong et al. [14] for

gradient stability in deep Transformers. An optional lightweight 3-layer Transformer encoder refines the CNN features before they reach the decoder's cross-attention heads.

Two architectural details proved critical for convergence on our Telugu dataset:

Embedding Scaling.

Scaling token embeddings by $\sqrt{d_{\text{model}}}$, as originally proposed by Vaswani et al. [1], proved necessary for our Telugu vocabulary:

$$\mathbf{e}_t = \text{Embed}(y_t) \cdot \sqrt{d_{\text{model}}} + \text{PE}(t) \tag{4}$$

Without this factor, embedding entries are initialised at considerably smaller magnitude than the sinusoidal positional encoding, and inspection of the resulting attention maps suggests the decoder attends based on position rather than token identity. We state this as an interpretation, not a measured causal mechanism. In either case, the failure mode is expected to be particularly damaging for Telugu, where word length varies widely and the same character recurs across many positions.

Weight Tying.

Following Press and Wolf [15], we tie the weight matrix between the input embedding layer and the output softmax projection:

$$P(y_t | y_{<t}, \mathbf{h}) = \text{softmax}(\mathbf{W}_{\text{embed}} \cdot \mathbf{d}_t) \tag{5}$$

This halves the embedding parameter count and forces each token vector to serve both as an input representation and an output predictor, acting as an implicit regulariser. The effect was consistent on our 91-token vocabulary.

Joint CTC-CE Training.

Training uses a weighted sum of cross-entropy and an auxiliary CTC loss computed on the encoder output:

$$\mathcal{L} = (1 - \lambda) \cdot \mathcal{L}_{\text{CE}} + \lambda \cdot \mathcal{L}_{\text{CTC}} \tag{6}$$

with $\lambda = 0.3$. The cross-entropy term includes label smoothing ($\epsilon = 0.05$, following Szegedy et al. [16]). The CTC branch encourages the encoder to produce features that are already roughly aligned with target characters, which in turn allows the attention decoder to converge faster and more reliably.

3.6 Telugu Linguistic Constraint Mask

Telugu, like most languages, has phonotactic constraints: not every character can follow every other character. We exploit this property by scanning the training transcripts, recording the character bigrams that actually occur, and storing the results in a binary matrix $\mathbf{M} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$:

$$\mathbf{M}[i][j] = \begin{cases} 1 & \text{if bigram } (c_i, c_j) \text{ observed in training} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Table 1 IIIT-HW-Telugu dataset statistics.

Split	Images	Vocab Size	Avg. Length
Train	80,693	87 chars	8.7 chars
Validation	20,048	–	–
Test	17,910	–	–
Total	118,651	91 tokens*	–

*Including 4 special tokens: PAD, SOS, EOS, UNK. All labels are NFC-normalised.

At each decoding step t, we look up the row corresponding to the most recently generated token c_{t-1} and subtract a large penalty $\delta = 10.0$ from the logits of any character that was never observed following c_{t-1} in the training data:

$$\ell'_j = \ell_j - \delta \cdot (1 - \mathbf{M}[c_{t-1}][j]) \tag{8}$$

We use a soft penalty rather than a hard mask deliberately, allowing the decoder to emit rare but potentially valid pairs that may simply not have appeared in the 80,000 training words. This design choice reflects a trade-off between the coverage of the training set and the true space of valid Telugu character sequences.

4 Experiments

4.1 Dataset

All experiments use the IIIT-HW-Telugu dataset [9, 10], a multi-writer collection of word-level handwritten Telugu images. Table 1 summarises the split sizes.

Each image is resized to 64×512 pixels, with padding applied to the shorter side to preserve the aspect ratio. Pixel values are normalised to the range $[-1, 1]$ using a mean of 0.5 and a standard deviation of 0.5. During training, we apply several on-the-fly augmentations to approximate the variability found in real handwriting: random rotation up to $\pm 5°$ (p=0.5), random perspective warping (p=0.3), brightness and contrast jitter, morphological dilation or erosion to simulate pen-thickness changes (p=0.3), and elastic distortion via Gaussian-smoothed displacement fields (α=10, σ=3, p=0.3). All augmentations run on the CPU during data loading, adding negligible GPU overhead. Fig. 2 illustrates the full preprocessing pipeline.

4.2 Implementation Details

Table 2 lists all training hyperparameters for both models.

Every experiment ran on a single NVIDIA GPU with 24 GB of VRAM, using PyTorch 2.7.1 and CUDA 11.8. The AR Transformer has 31.16 M trainable parameters, compared to approximately 12 M for the CTC baseline. Wall-clock training time was approximately 3.5 hours for the CTC model (50 epochs) and 5.4 hours for the

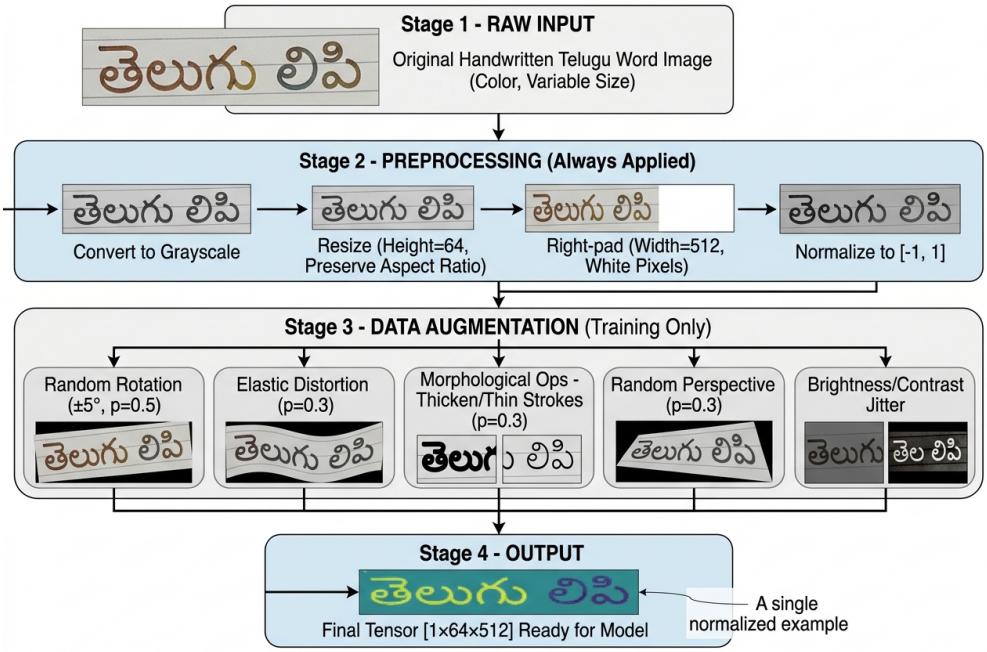


Fig. 2 Preprocessing and augmentation pipeline. Every image passes through fixed steps (grayscale conversion, resize, pad, normalise), and during training only, through a chain of random augmentations. Each augmentation is applied independently at the probability shown, so every epoch produces a slightly different version of every sample.

AR Transformer (80 epochs), with both models trained in FP16 mixed precision via PyTorch's built-in GradScaler.

Table 3 summarises the computational requirements of both models.

Inference latencies reported in Section 5 are per-image averages measured over the full test set at batch size 64 on the same GPU, and should be interpreted as relative rather than absolute figures: they vary by roughly an order of magnitude with batch size and data-loader settings.

4.3 Evaluation Protocol

We measure accuracy with two standard metrics. Character Error Rate (CER) is the total edit distance summed across all test samples, divided by the total number of ground-truth characters:

$$\text{CER} = \frac{\sum_{i=1}^{N} \text{ED}(\hat{y}_i, y_i)}{\sum_{i=1}^{N} |y_i|} \tag{9}$$

where $\text{ED}(\cdot, \cdot)$ denotes Levenshtein distance. Word Error Rate (WER) is the fraction of words where the prediction contains any error, however small. To quantify the uncertainty on each system's CER, we use non-parametric bootstrap resampling [18] over test samples with 1,000 resamples, and report 95% percentile confidence intervals. These are marginal intervals for each system evaluated independently; they are not a

Table 2 Training hyperparameters.

Parameter	CTC	AR Transformer
d_{model}	256	384
Encoder layers	–	3 (Transformer)
Decoder layers	2 (BiLSTM)	6 (Transformer)
Attention heads	–	8
d_{ff}	–	1,536
Batch size	64	64
Learning rate	3×10^{-4}	3×10^{-4}
LR schedule	OneCycleLR	Warmup + cosine
Warmup steps	implicit	4,000
Optimiser	AdamW	AdamW
Weight decay	10^{-4}	10^{-4}
Dropout	0.1	0.15
Label smoothing	–	0.05
CTC weight (λ)	–	0.3
Gradient clipping	5.0	5.0
Epochs	50	80
Mixed precision	FP16	FP16

Table 3 Computational comparison between the two models.

Metric	CTC	AR Transformer
Trainable parameters	∼12 M	31.16 M
Training epochs	50	80
Training time	∼3.5 h	∼5.4 h
Inference latency (ms/image)	0.3	1.4 (greedy)
Peak GPU memory	∼8 GB	∼14 GB

paired test of the difference between two systems, and we are careful not to interpret significance directly from whether they overlap.

5 Results and Analysis

5.1 Main Results

Table 4 lists CER, WER, and inference speed for every model variant we trained, all evaluated on the same 17,910 held-out test images.

We highlight three findings from these results (see also Figs. 3 and 4):

(1) The Transformer outperforms CTC, but convergence required specific architectural choices.

The best configuration (Row 4, beam-5, unconstrained) achieves 3.66% CER against the CTC baseline's 3.91%, a 6.4% relative reduction. The two bootstrap intervals abut rather than overlap ([3.54, 3.79] vs. [3.79, 4.04]), and the gap is consistent across all four Transformer variants and on both metrics, making a sampling artefact unlikely; a

Table 4 Test-set results on IIIT-HW-Telugu ($N = 17{,}910$). "Comp. CER" is CER restricted to the 10,129 compound words containing at least one virama. Speed is mean per-image inference latency at batch size 64.

#	Model	CER (%)	WER (%)	95% CI	Comp. CER (%)	Speed (ms)
1	CTC Baseline (BiLSTM)	3.91	24.80	[3.79, 4.04]	3.70	**0.3**
2	AR Transformer (greedy, unconstr.)	3.67	23.33	[3.55, 3.79]	3.32	1.4
3	AR Transformer (greedy, constr.)	3.74	23.32	[3.60, 3.88]	3.33	1.8
4	AR Transformer (beam-5, unconstr.)	**3.66**	23.34	[3.54, 3.79]	**3.31**	77.0
5	AR Transformer (beam-5, constr.)	3.67	**23.28**	[3.55, 3.79]	3.30	82.5

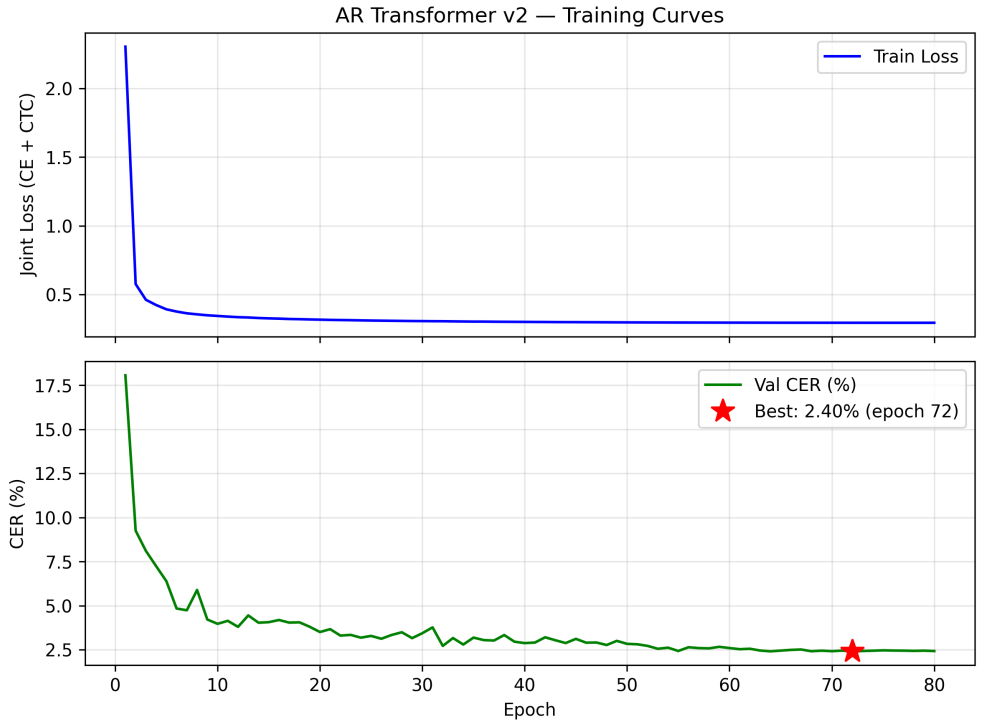


Fig. 3 AR Transformer training curves over 80 epochs. **Top:** the joint loss ($0.7\mathcal{L}_{\text{CE}} + 0.3\mathcal{L}_{\text{CTC}}$) drops steadily on the training set, while validation loss flattens around epoch 55. **Bottom:** validation CER reaches its minimum at 2.40% at epoch 72 ($\star$), after which mild overfitting begins.

paired bootstrap over per-sample differences would be the stricter test, and we regard the current evidence as suggestive rather than conclusive. We note that our initial Transformer runs produced CER above the CTC baseline; performance improved only after adding embedding scaling and weight tying (Section 3.5).

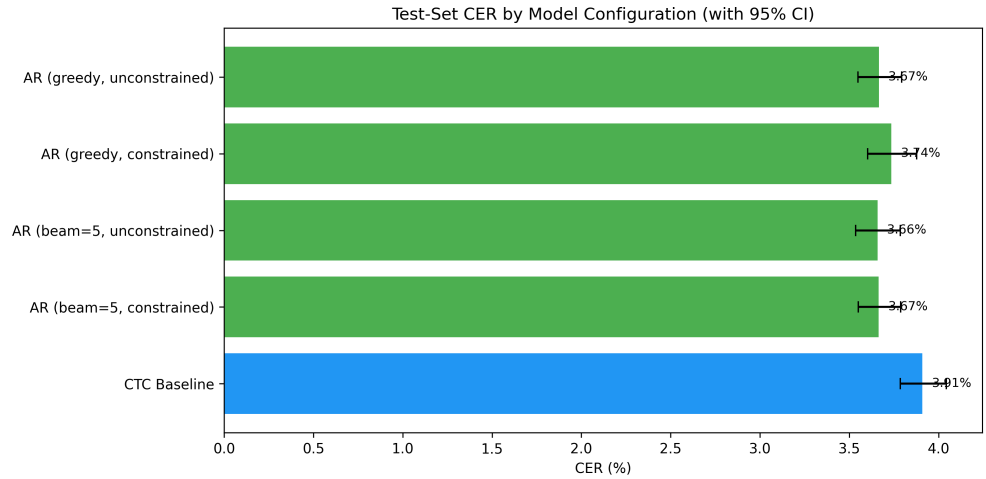


Fig. 4 Test-set CER for every model configuration. All four Transformer variants (rows 2–5) outperform the CTC baseline (row 1), though beam search adds only a marginal further gain over greedy decoding. Error bars show 95% bootstrap confidence intervals.

(2) Beam search yields negligible gains.

Greedy decoding achieves 3.67% CER; beam width 5 reduces this by only 0.01 percentage point. The confidence intervals overlap almost entirely, suggesting that encoder features, not the search strategy, are the binding constraint. Given the 55× increase in per-image latency, greedy decoding is the preferable operating point.

(3) The grammar mask produces mixed effects.

The Telugu transition matrix increases greedy CER from 3.67% to 3.74%, likely because it penalises rare-but-valid character pairs absent from the 80k training words. Constrained beam search, however, achieves the lowest WER across all configurations (23.28%), indicating that the mask suppresses some whole-word errors at the cost of additional character-level substitutions.

5.2 Comparison with Published Results

Table 5 compares our results with previously reported Telugu HTR numbers.

Among lexicon-free systems, our AR Transformer achieves the lowest CER on IIIT-HW-Telugu to our knowledge. The 3.66% CER represents a **20% relative reduction** from the 4.58% obtained by Dutta et al. [9] with IAM pre-training, and a **43% relative reduction** from the CRNN of Gongidi and Jawahar [10]. The only system reporting lower CER (1.52%) relies on a closed lexicon, restricting its output to previously seen words.

One caveat applies to the comparisons in Table 5. The prior numbers are quoted from their source papers, and we could not confirm that every one of them was computed on exactly the split of 17,910 test images we use here—Gongidi and Jawahar in particular report on the broader IIIT-INDIC-HW-WORDS collection. These

Table 5 Comparison with published Telugu HTR results.

Method	CER (%)	WER (%)	Lexicon
Dutta et al. [9] CNN-BLSTM	9.15	37.92	No
Dutta et al. [9] + IAM pretrain	4.58	23.98	No
Gongidi & Jawahar [10] CRNN	6.41	–	No
Gongidi & Jawahar [10] + Lexicon	1.52	2.84	**Yes**
Ours: CTC Baseline	3.91	24.80	No
Ours: AR Transformer	**3.66**	**23.28**	No

Table 6 Error rates by word type (AR = beam-5, unconstrained).

Word Type	n	Len. (chars)	CTC / AR CER (%)	CTC / AR WER (%)
Compound (virama)	10,129	9.87	3.70 / **3.31**	25.72 / **23.47**
Simple (no virama)	7,781	7.23	4.29 / 4.28	23.60 / **23.18**

comparisons should therefore be read as indicative of the range of published performance on handwritten Telugu words, not as controlled head-to-head measurements. The comparison between the final two rows—our own CTC baseline against our AR Transformer—is the one in this paper that is genuinely controlled, since both share an encoder, a split, and a training budget.

5.3 Compound vs. Simple Character Analysis

To localise each model's errors, we partition the test set into *compound* words containing at least one virama (n = 10,129) and *simple* words with no virama (n = 7,781). Table 6 reports CER and WER for each subset along with mean ground-truth length.

By CER alone, both models appear to perform *better* on compound words than on simple ones—a counter-intuitive result. One might conclude that conjunct glyphs, despite their visual complexity, are sufficiently distinctive to be recognised easily. The WER column contradicts this: at the word level, both models perform slightly *worse* on compound words (CTC 25.72% vs. 23.60%, AR 23.47% vs. 23.18%).

The discrepancy is a length effect. Compound words in this test set average 9.87 characters against 7.23 for simple words, roughly 36% longer. Because CER divides accumulated edit distance by total ground-truth characters, a longer word spreads the same word-level mistake over a larger denominator and so registers a lower CER. Normalising for this, the mean edit distance per erroneous word is nearly identical across the two subsets (1.39 characters for compound, 1.33 for simple). We flag this because subset CER comparisons are common in Indic HTR papers, and where the subsets differ systematically in word length—which is exactly what conditioning on the presence of a virama does—CER differences between them are not directly interpretable as differences in difficulty.

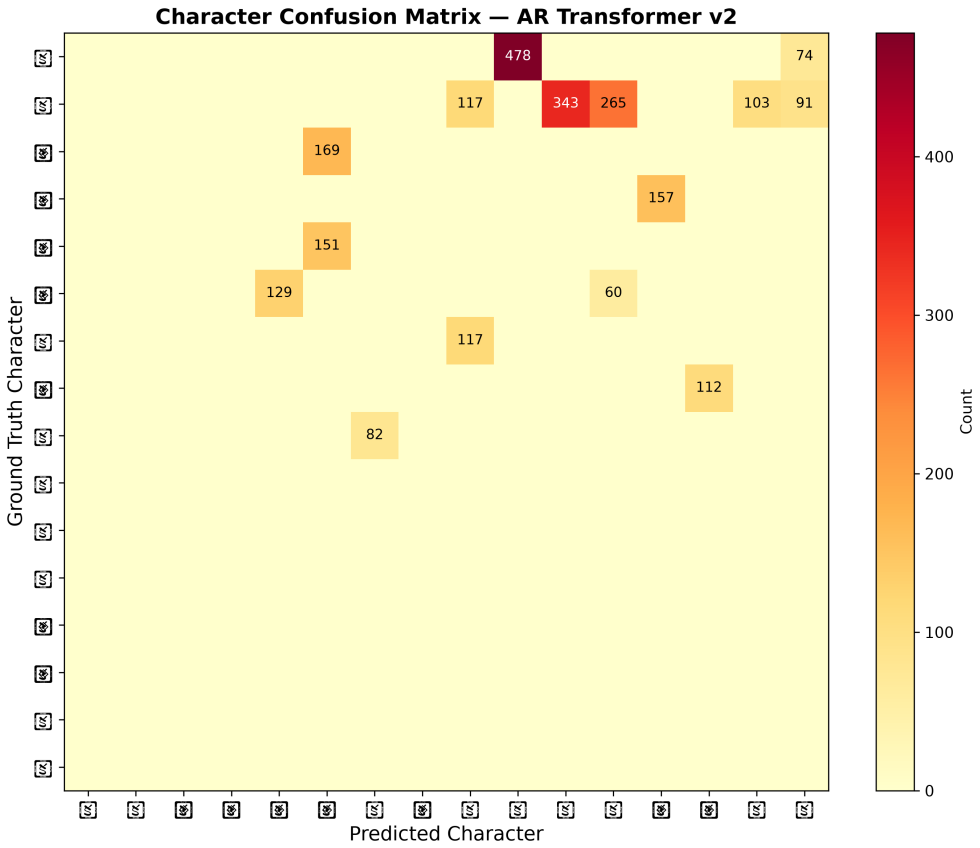


Fig. 5 Character-level confusion matrix for the AR Transformer on the test set. Darker cells indicate more frequent confusions. Most errors fall between consonant pairs that share stroke structure (e.g., *dda*↔*da*) and vowel signs that differ by a single stroke.

What the breakdown does show is *where* the Transformer earns its advantage. On compound words it reduces CER from 3.70% to 3.31%, a 10.5% relative reduction, and WER from 25.72% to 23.47%. On simple words the two models are indistinguishable (4.29% vs. 4.28% CER). Practically all of the Transformer's aggregate gain therefore comes from conjunct-bearing words, which is consistent with the hypothesis that modelling dependencies between adjacent characters helps most where those dependencies are structurally strongest.

5.4 Qualitative Error Analysis

The aggregate CER numbers mask an important distinction: the two architectures produce qualitatively different error patterns. Fig. 5 shows the Transformer's character confusion matrix; Fig. 6 presents representative predictions from both models.

CTC errors are predominantly *local substitutions*—one or two characters replaced by visually similar alternatives. The most frequent confusions involve consonant pairs



Fig. 6 Representative predictions from CTC and the AR Transformer. The top rows show cases both models recognise correctly; the bottom rows show typical failures. CTC tends to swap visually similar consonants, while the Transformer occasionally produces a fluent but entirely incorrect word.

with similar stroke geometry (*dda* ↔ *da*, 478 cases in our test set) and vowel-sign pairs differing by a single stroke (*e-matra* ↔ *i-matra*, 169 cases). Because each CTC prediction depends only on the encoder and not on prior outputs, these errors are distributed relatively uniformly across the test set and rarely corrupt more than two characters per word.

Transformer errors follow a different pattern. The decoder recognises most words correctly, accounting for its lower overall CER, but its failures tend to be severe. We refer to this failure mode as *autoregressive collapse*: the decoder confidently outputs a valid Telugu word that bears no relation to the input image. In one observed case, four visually distinct input images all produced the same common-word prediction. The mechanism is straightforward: if the first character is decoded incorrectly, the decoder's self-attention over its own output history reinforces the error, producing a coherent but entirely wrong character sequence. CTC is structurally immune to this failure mode because its predictions carry no feedback from prior outputs.

5.5 Validation-Test Gap

The Transformer's best validation CER was 2.40% (epoch 72), substantially lower than the 3.67% observed on the test set. We attribute this gap to three factors:

(1) Writer mismatch.

The test set likely contains handwriting styles underrepresented in the validation split. The Transformer, with more parameters than the CTC model, is more susceptible to overfitting to writer-specific characteristics.

(2) Checkpoint selection bias.

Selecting the checkpoint with the lowest validation CER from 80 epochs introduces a mild form of overfitting to the validation distribution.

(3) Limited training data.

Systems such as TrOCR [4] and PARSeq [5] train on millions of images with additional pre-training. Our 80k samples may be insufficient for the Transformer to generalise as broadly as it could with a larger corpus.

6 Discussion

The CTC-versus-Transformer comparison is less straightforward than the headline CER numbers indicate.

CTC has a structural advantage: it assumes left-to-right character order and that encoder outputs are already roughly aligned with them. This built-in prior acts as free regularisation and reduces training cost—CTC reaches 3.91% CER in 50 epochs (3.5 hours), whereas the Transformer requires 80 epochs (5.4 hours) to reach 3.66%. The Transformer must learn left-to-right alignment through cross-attention. We emphasise that this is an observation about training budget, not a controlled data-efficiency experiment: we did not sweep training-set size and cannot determine at what scale the two curves would cross.

The autoregressive collapse described in Section 5.4 has practical implications. In a pipeline digitising historical Telugu manuscripts or legal records, a model that occasionally produces a fluent but entirely incorrect word is more problematic than one that makes frequent small substitutions, because the errors are harder to detect during review. For such applications CTC may be the safer deployment choice, whereas the Transformer is preferable when minimising average CER is the primary objective.

The Telugu grammar mask did not improve CER. This is expected in retrospect: the transition matrix was built exclusively from training-set bigrams, so any valid character pair absent from those 80k words receives a penalty at test time. A mask derived from formal phonological rules or from a larger unlabelled text corpus would likely perform better.

6.1 Generalisability to Other Indic Scripts

Our experiments are limited to Telugu, but several findings are likely relevant to other Brahmic scripts. Kannada and Malayalam also feature extensive consonant conjuncts built through the virama, and the observation that the Transformer's advantage concentrates on conjunct-bearing words suggests autoregressive decoders may help in those scripts too. That said, each script has its own character-level ambiguities—for instance, the confusable consonant pairs differ across scripts—and the best model configuration will vary accordingly.

6.2 Limitations

Three limitations should be noted. First, our observation that embedding scaling and weight tying are necessary for convergence is based on our training runs rather than a controlled ablation with all other variables fixed; we report it as a practical engineering finding rather than a measured causal effect. A related ablation on an earlier architecture version removed the auxiliary CTC loss and saw CER rise from 4.89% to 5.43%, supporting the joint-loss design, though those numbers come from a different configuration and are not directly comparable to Table 4. Second, the 1.27-point gap between best validation CER (2.40%) and test CER (3.67%) is non-trivial, and because the IIIT-HW-Telugu splits are not writer-disjoint by construction, we cannot fully disentangle writer overfitting from checkpoint-selection bias. Third, our comparison against published systems is not controlled, for the reasons given in Section 5.2.

6.3 Future Work

Several directions follow from this work. Pre-training the Transformer decoder on synthetic Telugu text rendered from digital fonts could supply language-modelling knowledge without additional handwritten data. Non-autoregressive decoders such as PARSeq [5] may combine the Transformer's contextual modelling with CTC-style parallel decoding, avoiding exposure bias. An ensemble of CTC and Transformer outputs is also worth investigating, given that their error profiles are largely complementary. Finally, extending this comparison to other low-resource Indic scripts—particularly those with extensive conjunct systems such as Kannada and Malayalam—would help determine whether the patterns observed here are Telugu-specific or generalise across Brahmic scripts.

7 Conclusion

This paper compared CTC and autoregressive Transformer decoders for Telugu word-level handwriting recognition, using a shared stride-modified ResNet-18 encoder for a controlled comparison. The Transformer, trained with a joint CTC-attention loss, achieves 3.66% CER on the IIIT-HW-Telugu test set—to our knowledge the lowest lexicon-free result reported on this benchmark, and a 20% relative reduction from the 4.58% of Dutta et al. [9]. Two architectural details—embedding scaling by $\sqrt{d_{\text{model}}}$ and weight tying—were essential for convergence in our experiments. Decomposing results by word type revealed that the Transformer's advantage comes entirely

from conjunct-bearing words, and also that subset CER comparisons are confounded by systematic word-length differences, a point worth considering when interpreting compound-versus-simple results in Indic HTR. Finally, we observe a deployment trade-off: CTC produces frequent small character-level errors, while the Transformer is more accurate on average but susceptible to autoregressive collapse. Code and trained models are publicly available.

Data Availability

The IIIT-HW-Telugu dataset used in this study is publicly available from the Centre for Visual Information Technology (CVIT), IIIT Hyderabad, at https://cvit.iiit.ac.in/research/projects/cvit-projects/indic-hw-data. All source code, configuration files, trained model weights, and training scripts are released at https://github.com/Bhargav-252006/telugu_htr.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of Interest

The authors declare that they have no conflict of interest.

Author Contributions

Sakilam Bhargav: Conceptualisation, Methodology, Software, Investigation, Writing—Original Draft. **Dr. R. Venkata Ramana Chary**: Supervision, Review & Editing. **Thatha Nikitha**: Data Curation, Validation, Visualisation. **Sabavat Vinod Nayak**: Software, Testing, Review & Editing.

Acknowledgements

The authors thank the Centre for Visual Information Technology (CVIT) at IIIT Hyderabad for making the IIIT-HW-Telugu dataset publicly available.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 5998–6008 (2017)

[2] Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: International Conference on Machine Learning (ICML), pp. 369–376 (2006)

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

[4] Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., Wei, F.: TrOCR: Transformer-based optical character recognition with pre-trained models. In: AAAI Conference on Artificial Intelligence, vol. 37, pp. 13094–13102 (2023)

[5] Bautista, D., Atienza, R.: Scene text recognition with permuted autoregressive sequence models. In: European Conference on Computer Vision (ECCV), pp. 178–196 (2022)

[6] Shi, B., Bai, X., Yao, C.: 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 (2017)

[7] Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In: International Conference on Document Analysis and Recognition (ICDAR), pp. 67–72 (2017)

[8] Scheidl, H., Fiel, S., Sablatnig, R.: Word beam search: A connectionist temporal classification decoding algorithm. In: International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 253–258 (2018)

[9] Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.V.: Improving CNN-RNN hybrid networks for handwriting recognition. In: International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 80–85 (2018)

[10] Gongidi, S., Jawahar, C.V.: IIIT-INDIC-HW-WORDS: A dataset for Indic handwritten text recognition. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 444–459 (2021)

[11] Kasuba, B.V., Kudale, D., Subramanian, V., Chaudhuri, P., Ramakrishnan, G.: PLATTER: A page-level handwritten text recognition system for Indic scripts. arXiv preprint arXiv:2502.06172 (2025)

[12] Watanabe, S., Hori, T., Kim, S., Hershey, J.R., Hayashi, T.: Hybrid CTC/attention architecture for end-to-end speech recognition. IEEE Journal of Selected Topics in Signal Processing **11**(8), 1240–1253 (2017)

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

[14] Xiong, R., et al.: On layer normalization in the Transformer architecture. In: International Conference on Machine Learning (ICML), pp. 10524–10533 (2020)

[15] Press, O., Wolf, L.: Using the output embedding to improve language models. In: European Chapter of the ACL, pp. 157–163 (2017)

[16] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (2016)

[17] Eberhard, D.M., Simons, G.F., Fennig, C.D.: Ethnologue: Languages of the World, 27th edn. SIL International, Dallas (2024)

[18] Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall, New York (1993)

[19] Mondal, A., Jawahar, C.V.: ICDAR 2023 competition on handwriting recognition of Indic scripts. In: International Conference on Document Analysis and Recognition (ICDAR), pp. 460–475 (2023)

[20] Kang, L., Riba, P., Wang, Y., Rusiñol, M., Fornés, A., Villegas, M.: GANwriting: Content-conditioned generation of styled handwritten word images. In: European Conference on Computer Vision (ECCV), pp. 273–289 (2020)