Telugu Offline Handwriting Recognition with Confidence Calibration

Team Members:

Vetcha Pankaj Nath, 221188
Adithya Lingala, 220587

Vipul Arora

Contents

1. Introduction

Handwriting recognition (HWR) remains a challenging task in computer vision and document analysis, especially for complex scripts like Telugu. The intricate nature of Telugu characters, with their curved shapes and nuanced connections, presents unique challenges compared to Latin scripts.

While extensive research has focused on improving recognition accuracy for various scripts, including Indic languages, most existing approaches dont worry about confidence estimation—the ability of a system to reliably assess when its predictions might be incorrect. This is particularly important in practical applications such as document digitization, educational assessment, and heritage manuscript preservation, where knowing the reliability of a prediction could be as valuable as the prediction itself.

This work addresses two critical aspects of handwritten text recognition for Telugu script:

1. Accurate recognition using a Convolutional Recurrent Neural Network (CRNN) architecture
2. Reliable confidence estimation for predictions through multiple calibration techniques

The primary contributions of this work include:

- Implementation and evaluation of a CRNN model with CTC loss for Telugu handwriting recognition recognition:
- Development and integration of comprehensive confidence calibration methods:
  - Temperature scaling for improved confidence estimates
  - Monte Carlo Dropout for uncertainty estimation
  - Step-dependent temperature scaling for position-specific calibration
- Thorough evaluation and analysis of both recognition performance and confidence calibration quality on Telugu handwritten text.

2. Methodology

2.1. Dataset and Preprocessing

2.1.1. Telugu Handwriting Dataset

For our experiments, we utilize the Telugu portion of the IIIT-INDIC-HW-WORDS dataset, which is one of the most comprehensive collections for Indic handwritten text recognition. The dataset contains approximately 120,000 word instances written by 11 writers and encompasses a lexicon size of 12,945 unique words. This scale makes it comparable to widely-used datasets for Latin scripts like IAM.
The dataset statistics are summarized in Table 1:

Table 1. Dataset Statistics

Split	Word Instances	Unique Words	Unique Characters
Training	80,637	10,356	67
Validation	19,980	5,482	67
Test	17,898	5,189	67
Total	118,515	12,945	67

Telugu script features complex character structures with curved shapes and numerous possible character combinations. Words in Telugu tend to be longer compared to Indo-Aryan languages, with an average word length of approximately 9 characters.

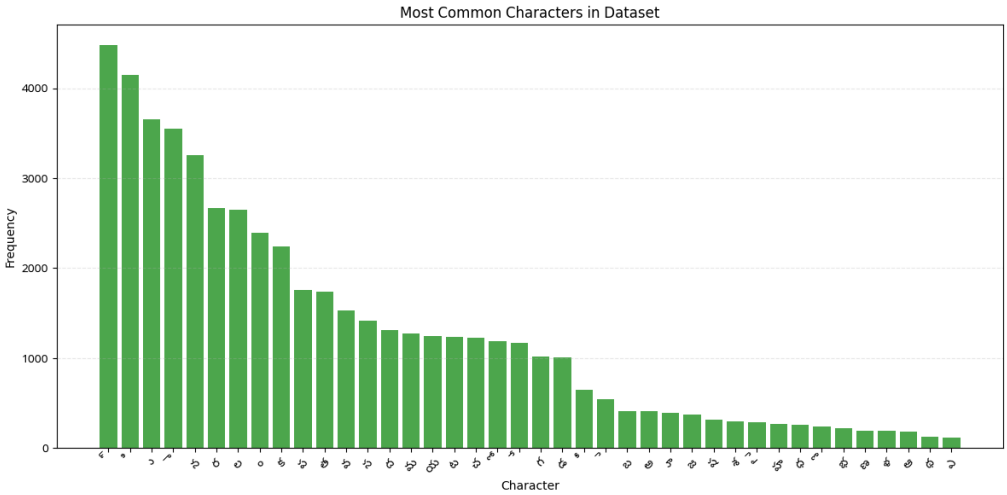


Figure 1. Distribution of character frequencies in the dataset, showing the most common Telugu characters.

2.1.2. Preprocessing Pipeline

We implemented a comprehensive preprocessing pipeline to prepare the images for model training and evaluation:

1. **Resizing with Aspect Ratio Preservation**: All images are resized to a fixed height (64 pixels) while maintaining the original aspect ratio to preserve character proportions.
2. **Grayscale Conversion**: All images are converted to grayscale to reduce computational complexity and focus on structural information rather than color.
3. **Normalization**: Pixel values are normalized to the range [-1, 1] to improve training stability and convergence.

For data augmentation during training, we employed the following techniques to enhance model robustness:

1. **Random Affine Transformations**: Small rotations ($\pm 3°$) and shearing ($\pm 5\%$) to simulate natural handwriting variations and improve resilience to writing angle variations.
2. **Elastic Deformations**: Controlled warping of the images to mimic the natural variations in handwriting strokes and pen pressure.
3. **Brightness and Contrast Adjustment**: Random adjustments to simulate different scanning and lighting conditions that might be encountered in real-world documents.

These preprocessing and augmentation techniques were crucial for enhancing the generalization capability of our models across different writing styles and image quality conditions.

2.2. Model Architectures

2.2.1. CRNN with CTC Loss

The Convolutional Recurrent Neural Network (CRNN) architecture combines the strengths of CNNs for feature extraction and RNNs for sequence modeling. Our implementation follows the established frameworks for sequence recognition with three main components:

1. **Convolutional Feature Extraction**: A series of convolutional layers extract visual features from the input image with the following structure:
    - 5 convolutional blocks with increasing channel dimensions ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512 \rightarrow 512$)
    - Each block contains convolution, batch normalization, and ReLU activation

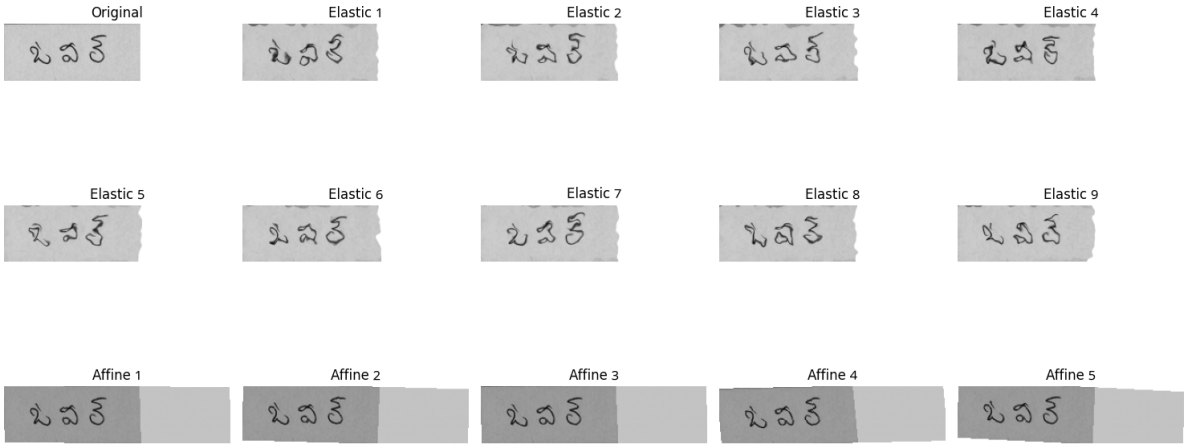


Figure 2. Sample images after data augmentation

- Pooling layers to reduce spatial dimensions (2×2 max pooling in the first two blocks, 1×2 max pooling in the subsequent blocks to preserve horizontal information)

2. **Recurrent Sequence Modeling**: Bidirectional LSTM layers process the extracted features as a sequence:
   - Features from CNN are reshaped into a sequence of feature vectors (width × feature_dim)
   - Two BiLSTM layers with 256 hidden units each for capturing contextual information
   - Dropout (0.2) between LSTM layers for regularization
3. **Transcription Layer**: Projects the LSTM outputs to character probabilities:
   - Linear projection to vocabulary size plus one (for blank token required by CTC)
   - Softmax activation for character probabilities at each timestep

The model is trained using Connectionist Temporal Classification (CTC) loss, which enables alignment-free sequence learning by automatically finding the alignment between the input image sequence and output text. This eliminates the need for explicit character segmentation, which is particularly advantageous for scripts like Telugu where character boundaries can be ambiguous.

During inference, the model outputs a probability distribution over characters for each time step. The final transcription is obtained using CTC decoding, which removes duplicate consecutive characters and blank tokens.

2.3. Confidence Calibration Methods

deep neural networks, while achieving high accuracy, often produce poorly calibrated confidence scores—they tend to be overconfident in their predictions, even when wrong. A well-calibrated model should produce confidence scores that reflect the actual probability of correctness. For example, among all predictions with 80% confidence, approximately 80% should be correct.

We implemented and evaluated several confidence calibration methods:

2.3.1. Temperature Scaling

Temperature scaling is a simple yet effective post-processing technique for calibrating neural network outputs. It involves dividing the logits (pre-softmax activations) by a temperature parameter τ before applying the softmax function:

$$p_i(x;\tau) = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)} \tag{1}$$

where z_i are the logits and $\tau > 0$ is the temperature parameter. When $\tau > 1$, the probability distribution becomes more uniform (reducing confidence), and when $\tau < 1$, it becomes more peaked (increasing confidence).

The optimal temperature is determined on a validation set by minimizing the Expected Calibration Error (ECE). The calibration process does not affect the model's accuracy, as the relative ordering of logits remains unchanged.

For our implementation, we extended basic temperature scaling to handle sequence outputs by:

1. Finding the optimal temperature value that minimizes calibration error on the validation set
2. Calculating word-level confidence scores as the geometric mean of character-level confidences
3. Applying the same temperature across all characters in the CRNN model

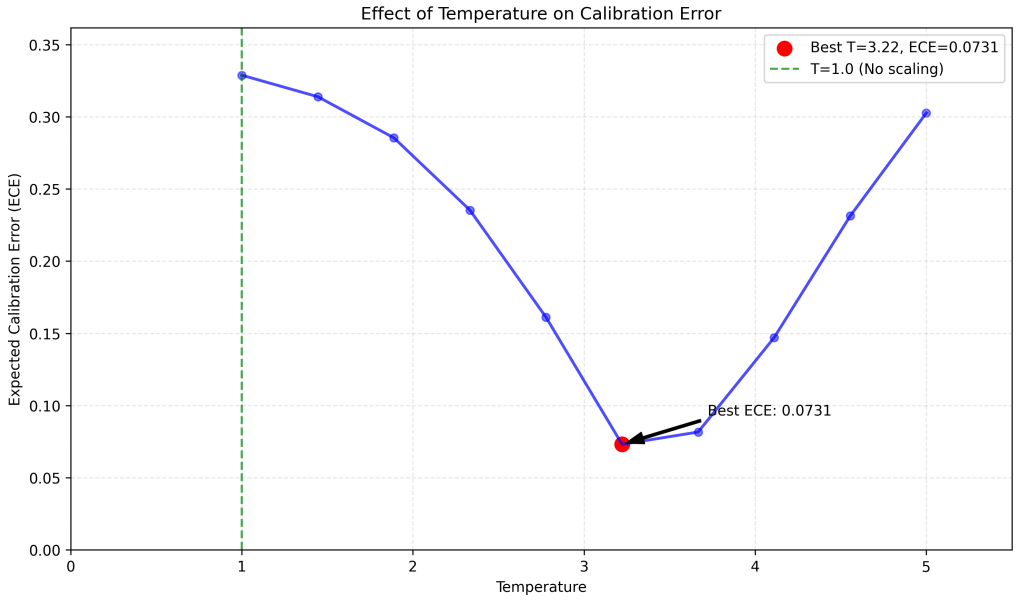


Figure 3. Effect of Temperature.

2.3.2. Monte Carlo Dropout

Monte Carlo Dropout provides a scalable approximation to Bayesian inference by performing stochastic forward passes with dropout at both training *and* inference time. The approach involves:

1. Enabling dropout during inference (not just training)
2. Performing T stochastic forward passes with different dropout masks
3. Aggregating the sampled outputs to estimate predictive statistics

We distinguish between two types of predictive uncertainty:

- **Epistemic uncertainty**: uncertainty in the model parameters due to limited data or knowledge, which can be reduced with more data
- **Aleatoric uncertainty**: irreducible noise inherent in the data (e.g., sensor noise, label noise)

Estimating Epistemic Uncertainty By keeping dropout active at inference and collecting T samples

$$\{\,\hat{y}_t = f_{W_t}(\mathbf{x})\}_{t=1}^{T},$$

we compute the sample mean

$$\bar{y} \;=\; \frac{1}{T}\sum_{t=1}^{T}\hat{y}_t$$

and the sample variance

$$\mathrm{Var}_{\mathrm{epi}} \;=\; \frac{1}{T}\sum_{t=1}^{T}\big(\hat{y}_t - \bar{y}\big)^2,$$

which serves as the epistemic uncertainty estimate.

Implementation for Sequence Recognition

1. Keep dropout layers active during inference.
2. Perform T stochastic forward passes.
3. Compute $\bar{y}$ as the final prediction (e.g., via beam search or most frequent output).
4. Estimate epistemic uncertainty as $\mathrm{Var}_{\mathrm{epi}} = \frac{1}{T}\sum_t(\hat{y}_t - \bar{y})^2$.

2.3.3. Step-Dependent Temperature Scaling

Recognizing that different positions in a sequence may require different calibration, we implemented Step-Dependent Temperature Scaling (STS). This method assigns different temperature values to different positions in the sequence:

$$p_i(x_t;\tau_t) = \frac{\exp(z_i^t/\tau_t)}{\sum_j \exp(z_j^t/\tau_t)} \tag{2}$$

where τ_t is the temperature for position t in the sequence.
To avoid overfitting, we used a parameter-sharing scheme:

1. Unique temperature values for the first k positions
2. A shared temperature for all positions beyond k

This approach recognizes that early positions in the sequence may have different confidence characteristics than later positions, which is particularly relevant for Sequential models where early errors can propagate to later positions.

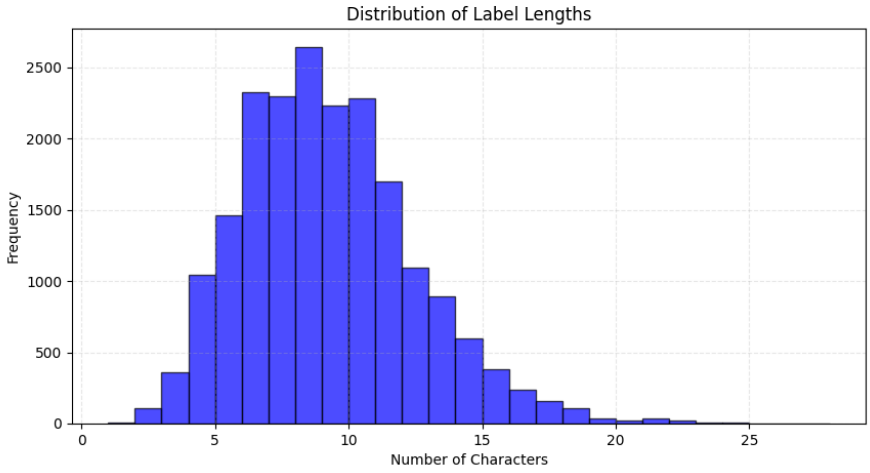


Figure 4. Word length distribution

2.4. Confidence Aggregation Methods

For sequence-based tasks like handwriting recognition, we need a method to combine character-level confidences into a word-level confidence score. We explored three aggregation strategies:

1. **Geometric Mean**: Calculates the n^{th} root of the product of n probabilities, providing a length-normalized confidence:
$$\text{confidence} = \left(\prod_i p_i \right)^{1/n}$$
2. **Product**: Simply multiplies all character probabilities :
$$\text{confidence} = \prod_i p_i$$
3. **Minimum**: Takes the minimum probability as the sequence confidence :
$$\text{confidence} = \min_i (p_i)$$

3. Experiments

3.1. Experimental Setup

3.1.1. Training Protocol

Both CRNN and PARSeq models were trained using the following protocols:

CRNN Training:

- Optimizer: Adam with initial learning rate of 0.001
- Batch size: 32
- Learning rate scheduling: ReduceLROnPlateau with factor 0.5 and patience 3
- Epochs: 20 (with early stopping)
- Loss function: CTC loss
- Gradient clipping: 5.0

we used early stopping based on validation character error rate (CER) with a patience of 5 epochs. The best model was selected based on validation CER.

3.1.2. Evaluation Metrics

We evaluated the models using the following metrics:

Recognition Performance:

1. **Character Error Rate (CER)**: The percentage of characters that need to be inserted, deleted, or substituted to convert the predicted text to the ground truth:
$$\text{CER} = \frac{\text{Levenshtein distance}(pred, target)}{\text{length}(target)} \times 100\% \tag{3}$$
2. **Word Error Rate (WER)**: The percentage of words that are incorrectly predicted:
$$\text{WER} = \frac{\text{Number of incorrect words}}{\text{Total number of words}} \times 100\% \tag{4}$$

Confidence Calibration:

1. **Expected Calibration Error (ECE)**: Measures the difference between confidence and accuracy across M bins:
$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{5}$$
where B_m is the m-th bin, $acc(B_m)$ is the accuracy in that bin, and $conf(B_m)$ is the average confidence in that bin.
2. **Maximum Calibration Error (MCE)**: Measures the worst-case deviation between confidence and accuracy across M bins:
$$\text{MCE} = \max_{1 \le m \le M} |acc(B_m) - conf(B_m)| \tag{6}$$
where B_m is the m-th bin, $acc(B_m)$ is the accuracy in that bin, and $conf(B_m)$ is the average confidence in that bin.
3. **Brier Score**: Mean squared error between probabilities and actual outcomes:
$$\text{Brier Score} = \frac{1}{N} \sum_{i=1}^{N} (conf_i - correct_i)^2 \tag{7}$$

3.1.3. Implementation Details

The system was implemented in PyTorch with the following specifications:

- Training hardware: NVIDIA GeForce RTX 3050 GPU (4GB memory)
- All confidence calibration methods were implemented as post-processing techniques that can be applied to any pre-trained model without retraining

3.2. Recognition Results

Table 2 presents the recognition performance of both CRNN and PARSeq models on the Telugu handwriting dataset:

Table 2. Recognition performance of CRNN model

Model	CER (%)	WER (%)
CRNN	5.51	31.76

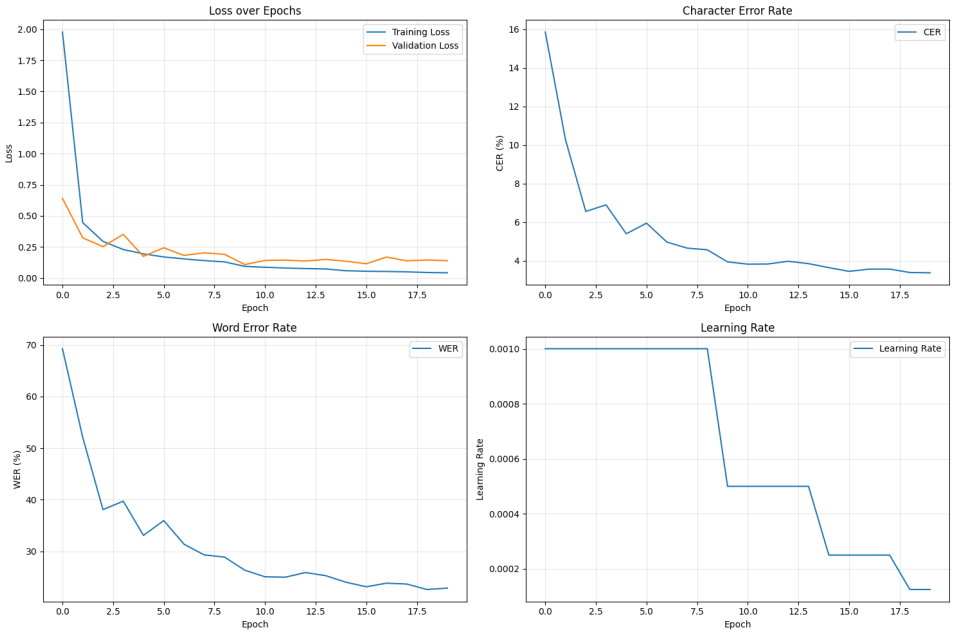


Figure 5. Training History of CRNN model

3.3. Confidence Calibration Results

To assess the effectiveness of confidence calibration, we first analyzed the calibration of the base model without any calibration techniques, then applied our different methods and evaluated them with varying aggregation strategies.

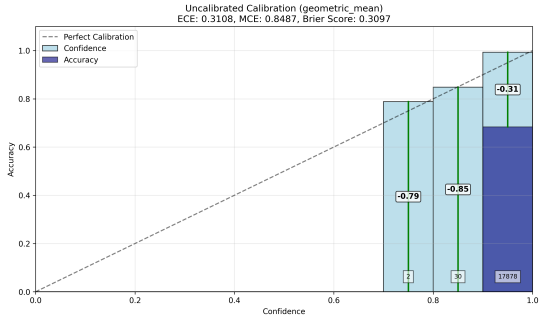
3.3.1. Geometric Mean Aggregation

Table 3 summarizes the calibration metrics with geometric mean aggregation:

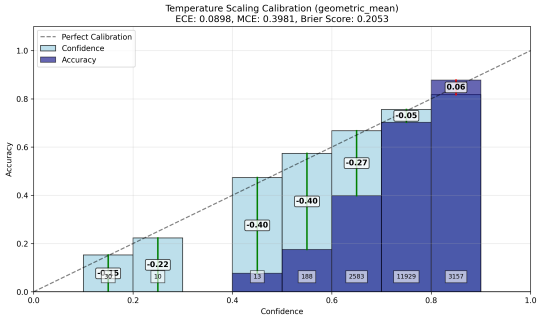
Table 3. Calibration metrics with geometric mean aggregation

Model	Calibration Method	ECE (%)	MCE (%)	Brier Score
CRNN	Uncalibrated	31.08	84.87	0.310
CRNN	Temperature Scaling	8.98	39.81	0.205
CRNN	STS	**7.91**	**39.64**	**0.204**
CRNN	MC Dropout	23.37	47.05	0.230

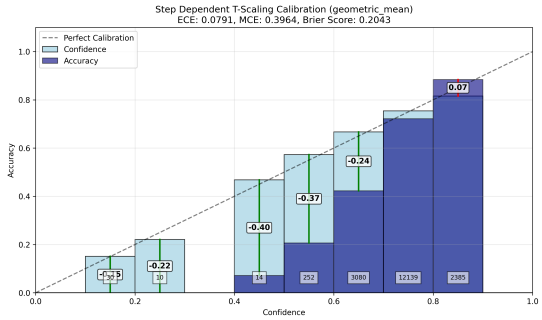
Figure ?? shows reliability diagrams for uncalibrated, temperature scaling, and step-dependent temperature scaling methods with Geometric Aggregation.



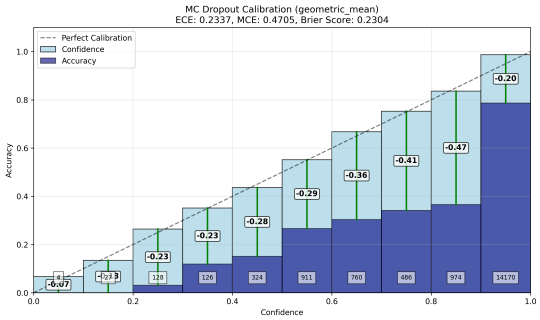
(a) Uncalibrated



(b) T scaling



(c) STS scaling



(d) MC Dropout

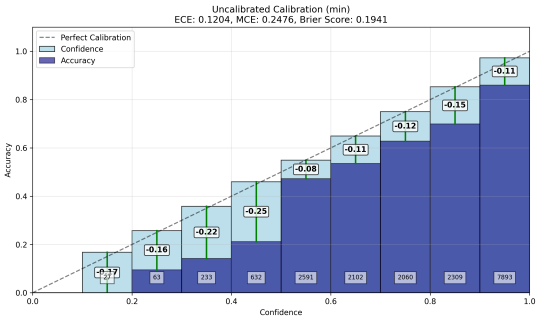
3.3.2. Minimum Aggregation

Table 4 summarizes the calibration metrics with Minimum Aggregation:

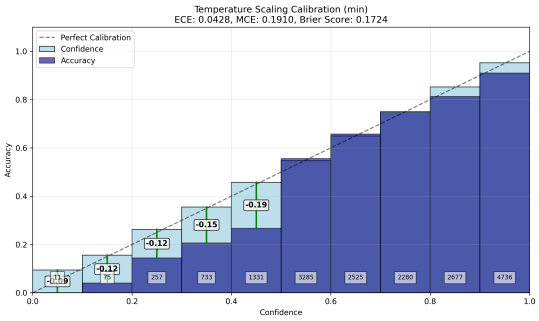
Table 4. Calibration metrics with minimum aggregation

Model	Calibration Method	ECE (%)	MCE (%)	Brier Score
CRNN	Uncalibrated	12.04	24.76	0.194
CRNN	Temperature Scaling	4.28	19.10	**0.172**
CRNN	STS	**3.96**	**18.16**	0.173
CRNN	MC Dropout	7.38	22.40	0.187

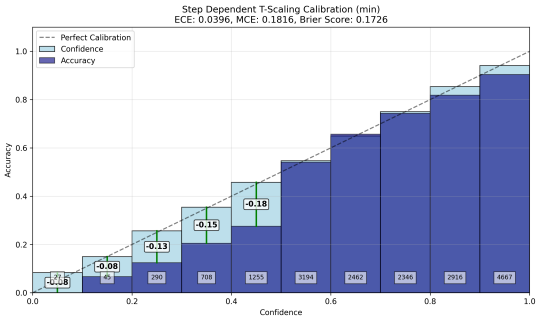
Figure ?? shows reliability diagrams for uncalibrated, temperature scaling, and step-dependent temperature scaling methods with minimum aggregation.



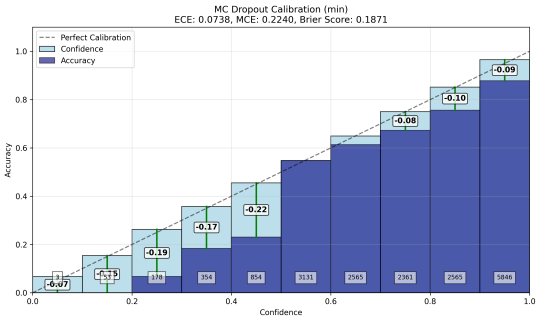
(a) Uncalibrated



(b) T scaling



(c) STS scaling



(d) MC Dropout

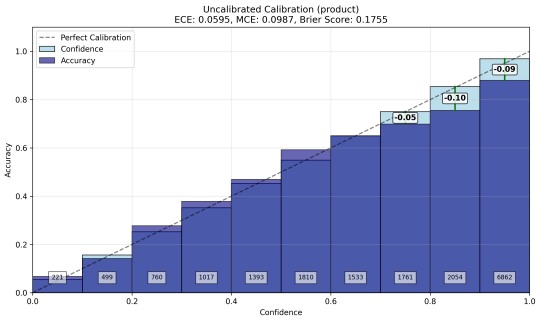
3.3.3. Product Aggregation

Table 5 summarizes the calibration metrics with Product Aggregation:

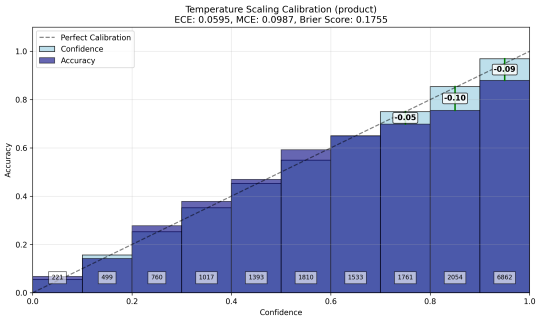
Table 5. Calibration metrics with product aggregation

Model	Calibration Method	ECE (%)	MCE (%)	Brier Score
CRNN	Uncalibrated	5.95	9.87	0.176
CRNN	Temperature Scaling	**5.95**	**9.87**	**0.176**
CRNN	STS	9.23	15.58	0.184
CRNN	MC Dropout	8.87	17.94	0.186

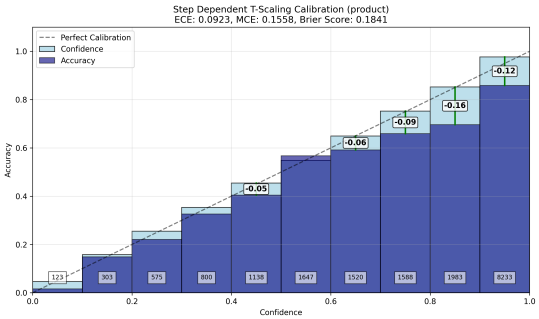
Figure ?? shows reliability diagrams for uncalibrated, temperature scaling, and step-dependent temperature scaling methods with Product Aggregation.



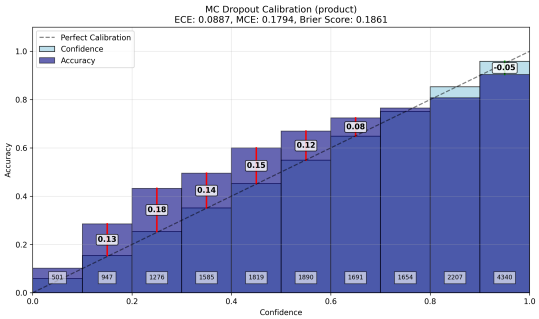
(a) Uncalibrated



(b) T scaling



(c) STS scaling



(d) MC Dropout

4. Results and Discussion

4.1. Recognition Performance

The CRNN model achieved a Character Error Rate (CER) of 5.31% and a Word Error Rate (WER) of 31.76% on the test set. This discrepancy between CER and WER indicates that while the model makes relatively few character-level errors, these errors often result in incorrectly recognized words, which is typical for languages with longer average word lengths like Telugu.

Analysis of the error patterns revealed that most errors occur with less common characters and similar-looking character pairs. The model also struggled more with longer words and words containing conjunct consonants.

4.2. Confidence Calibration Analysis

Our investigation into confidence calibration methods yielded several significant insights:

1. **Effectiveness of Temperature Scaling**: Simple temperature scaling substantially improved calibration, reducing ECE from 31.08% to 8.98% with geometric mean aggregation, and from 12.04% to 4.28% with minimum aggregation. This suggests that much of the miscalibration in the base model can be addressed with a single global parameter.
2. **Step-Dependent Temperature Scaling**: This method consistently outperformed uniform temperature scaling, especially with minimum aggregation, reducing ECE further to 3.96%. This confirms our hypothesis that different positions in the sequence require different calibration adjustments.
3. **Aggregation Methods Matter**: The choice of aggregation method significantly impacts calibration quality. Minimum aggregation provided the best calibration results across all methods, while product aggregation led to the lowest Brier scores but higher ECE for some methods.
4. **Monte Carlo Dropout Performance**: While MC Dropout provided uncertainty estimates that other methods do not, it showed mixed calibration performance, performing better than uncalibrated but worse than temperature scaling methods in most cases.

It is worth noting that with product aggregation, the uncalibrated and temperature-scaled models showed identical performance metrics.

4.3. Best Practices for Confidence Estimation

Based on our experimental results, we suggest the following best practices for confidence estimation in Telugu handwriting recognition:

1. **Minimum Aggregation is Preferred**: Using the minimum character confidence as the word-level confidence provides the best calibration across methods.
2. **Step-Dependent Temperature Scaling**: This method achieves the best calibration, especially with minimum aggregation, although it requires more parameters and optimization.
3. **Temperature Selection**: The optimal temperature depends on the aggregation method used. For minimum aggregation, a temperature of around 1.6 worked best, while for geometric mean aggregation, a higher temperature of approximately 3.9 was optimal.
4. **Balancing Calibration Metrics**: While Expected Calibration Error (ECE) is the most commonly used metric, considering Maximum Calibration Error (MCE) and Brier Score provides a more comprehensive evaluation. Step-Dependent Temperature Scaling had the lowest ECE, but not always the lowest Brier score.

4.4. Practical Applications

The improved confidence calibration has several practical applications for Telugu handwriting recognition:

1. **Active Learning**: The uncertainty estimates can guide the selection of samples for annotation in an active learning framework, focusing annotation efforts on the most informative examples and potentially reducing data collection costs.
2. **Rejection Option**: In critical applications where errors can have significant consequences, the system can refuse to make a prediction when the confidence is below a threshold, ensuring a controlled error rate in the automated decisions.

5. Conclusion

In this work, we presented a comprehensive approach to Telugu handwriting recognition with confidence calibration. We demonstrated that:

1. CRNN with CTC loss provides effective recognition of Telugu handwritten text, achieving a Character Error Rate (CER) of 5.31%.
2. Neural networks are inherently miscalibrated, often producing overconfident predictions. However, these can be significantly improved with proper calibration techniques.
3. Step-Dependent Temperature Scaling offers the best calibration performance, reducing ECE to as low as 3.96% with minimum aggregation.
4. The choice of aggregation method for converting character-level to word-level confidence is crucial, with minimum aggregation providing the best results for calibration quality.

Our experimental results provide valuable insights into confidence calibration for sequence recognition tasks and establish a strong baseline for future work in Telugu handwriting recognition.

Thank You