

Beyond isolated characters: the first segmentation-free recogniser of handwritten Meitei Mayek words

Chingkheinganba Raj Kumar

u3647772@connect.hku.hk

The University of Hong Kong

Research Article

Keywords: Meitei Mayek, Manipuri, handwritten word recognition, synthetic training data, connectionist temporal classification, low-resource scripts

Posted Date: October 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11184324/v1>

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

[Read Full License](#)

Additional Declarations: No competing interests reported.

Beyond isolated characters: the first segmentation-free recogniser of handwritten Meitei Mayek words

Chingkheinganba Rajkumar ^{1*}

^{1*}School of Computing and Data Science, The University of Hong Kong, Hong Kong SAR, China.

Corresponding author(s). E-mail(s): chingkheinganbaofficial@gmail.com;

Abstract

Handwritten Meitei Mayek, the script of Manipuri, has so far been recognised one character at a time, although its most frequent errors involve characters whose identity depends on their position in a word. We present the first segmentation-free recogniser of handwritten Meitei Mayek words. Without transcribed handwritten words, we compose training words from the isolated characters of the Tezpur University Meitei Mayek Handwritten Character Database (TUMMHCD), restoring what the dataset lost: positions from a font, sizes from stroke thickness and spacing from real handwriting, with vowel signs kept in their syllables. A ConvNeXt-T encoder initialised from a TUMMHCD character network, a bidirectional long short-term memory (LSTM) network and connectionist temporal classification (CTC) read the words, with a character 6-gram language model. On 5,000 synthetic test words the recogniser has a character error rate of 0.24% and a word error rate of 1.57%; an ensemble of character classifiers given perfect segmentation, perfect zones and the same language model reaches 4.26% and 20.98%. Typed text (1,454,510 words) shows that the most frequent confusion, $\mathfrak{S}$ against $\mathfrak{S}$, is a spelling convention, not a visual distinction. On 100 unseen words handwritten on a tablet by a native writer, the character error rate is 2.96% and the word error rate 16.0%; 99% of the character errors involve the three signs that print places above the letter, which this writer places beside it. The code, models, real word set and a browser demonstration are publicly available.

Keywords: Meitei Mayek, Manipuri, handwritten word recognition, synthetic training data, connectionist temporal classification, low-resource scripts

Acknowledgements: The author thanks the creators of TUMMHCD for making the dataset public.

1 Introduction

Meitei Mayek (also written Meetei Mayek) is the script of Manipuri (Meiteilon), an official language of the Indian state of Manipur, and has its own Unicode block, U+ABC0 to U+ABFF [1]. Research on handwriting recognition for the script has so far concentrated on isolated characters. Several datasets of character images have been collected [2–4], and classifiers trained on them now reach high accuracy: the best reported result, 98.12% on the official test set of the Tezpur University Meitei Mayek Handwritten Character Database (TUMMHCD) [2], is obtained by an ensemble of six convolutional neural networks (CNNs) [5]. Handwriting, however, consists of words rather than isolated characters.

Many of the errors that remain in character classification cannot be corrected from the character image alone. In the ensemble of [5], four pairs of classes account for 148 of the 241 errors on the TUMMHCD test set. Three of these pairs look alike once a character is cropped and scaled to fill its frame, but they occupy different positions in a word: the vowel sign ◌◌ (yetnap) and the digit 0, the vowel sign ◌◌ (unap) and the letter ◌ (sam), and the letters ◌ (dil) and ◌ (ghou). The fourth pair, ◌ (i lonsum) and ◌ (i), which alone accounts for 78 errors, cannot be separated from isolated images at all: a two-class network reaches 68.2%, against a majority baseline of 67.1% [5]. A recogniser that reads whole words can exploit the position of each sign and the vocabulary of the language; one that reads isolated characters cannot.

To our knowledge, no published work recognises handwritten Meitei Mayek words or text lines end to end. Previous work segments handwritten pages into lines and words without recognising them [3], or corrects the output of a character classifier on segmented words using the zone of each character and the orthographic rules of the script [6, 7]. The only large dataset of handwritten Manipuri words, IIIT-Indic-HW-UC [8], is written in Bengali script. Recognition of Meitei Mayek text is available for print [9], and work on scene text detects the script without reading it [10].

The main obstacle is the lack of data: no transcribed images of handwritten Meitei Mayek words are available for training. Our approach is to compose training words from isolated characters and to evaluate the recogniser on real handwriting. The main contributions of this paper are as follows.

1. The first segmentation-free recogniser of handwritten Meitei Mayek words, which combines a convolutional encoder initialised from a network trained on isolated characters, a bidirectional long short-term memory (LSTM) network, connectionist temporal classification (CTC) and a character language model (Sections 6 and 8).
2. A method for composing realistic synthetic handwritten words from TUMMHCD characters at scale, which recovers the sizes and positions lost by the dataset’s images from a font and from the images themselves, and matches letter spacing to real handwriting (Section 5).
3. A test set of 100 real handwritten Meitei Mayek words, to our knowledge the first to be publicly released: written by a native writer on a tablet, text-disjoint from the training data, with a fixed evaluation protocol and the tools to extend it, under CC BY 4.0 (Section 9).

4. Evidence that $\mathfrak{S}$ against $\mathfrak{S}$, the largest source of error in isolated-character recognition, is a spelling convention rather than a visual distinction, so that a recogniser should read a single letter and render either spelling (Section 3).

In addition, we measure where the writer of the real set places the signs that print places above a letter, and extend the synthesiser to reproduce this placement; training with it is left to future work (Section 9.4).

2 Related work

Meitei Mayek recognition.

Datasets of handwritten Meitei Mayek characters include TUMMHCD, with 85,124 images of 55 classes [2], Mayek27 [3] and the collection of Nongmeikapam et al. [4], and convolutional classifiers achieve high accuracy on their isolated characters [5, 11]. Inunganbi et al. [3] segment 189 handwritten pages into lines and words, segmenting 88.96% of the words correctly, but classify isolated letters only. Hijam and Saharia [6] add a second stage to a character classifier applied to segmented words, in which the zone of a character (above, on or below the line) and orthographic rules restrict its class; on 100 handwritten words this raises character accuracy from 88.50% to 91.86%. Hijam’s thesis [7] combines a CNN character classifier with a character-level LSTM language model on word images and states the rule for $\mathfrak{S}$ and $\mathfrak{S}$ that we examine in Section 3. All of these systems segment first and classify characters one at a time, whereas our recogniser reads the word image as a whole. For printed text, NE-OCR [9] recognises word and line images in ten languages of Northeast India, Meitei Mayek among them, with a recogniser trained on rendered text, and EMBiL [10] detects Meitei Mayek and English scene text and identifies its language.

Indic handwritten text recognition.

Word-level recognition of Indic handwriting has progressed from zone-wise segmentation with hidden Markov models [12] to convolutional recurrent neural networks (CRNNs) trained with CTC [13, 14] on large word datasets for scripts such as Devanagari, Bengali and Telugu [8, 14]. IIIT-Indic-HW-UC [8] contains 200,000 handwritten Manipuri words by 101 writers, all in Bengali script, which a CRNN baseline reads with 90.99% character accuracy and 83.38% word accuracy; its split is not stated to be writer- or text-disjoint. Our real set is small, but it is text-disjoint from the training data by construction, and its writer is not among the writers of the training characters.

Synthetic handwriting.

Where transcribed handwriting is scarce, recognisers are trained on rendered or synthesised text, as in scene-text recognition [15]. Approaches include handwriting-style fonts with deformations [16, 17], also used for Devanagari and Bengali words [18], concatenations of character images, and generative models of styled handwriting [19–24]. The gap between synthetic and real handwriting is well documented [25]. Meitei Mayek has neither a handwriting font nor a generative model of its handwriting,

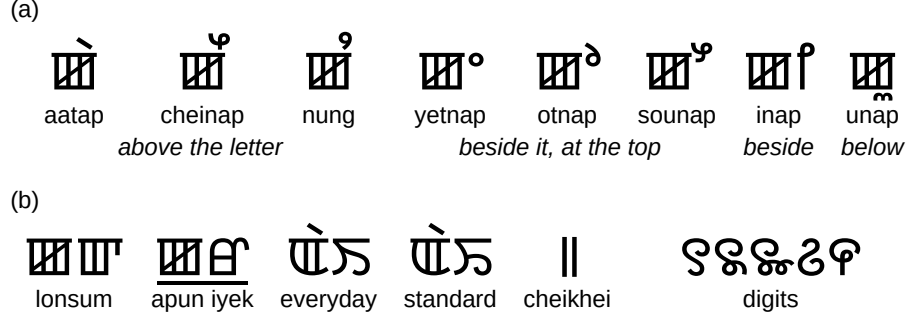


Fig. 1 Meitei Mayek in print (Noto Sans Meitei Mayek [26]). (a) The letter 𑜋 (kok) with each vowel sign and nung; print places ̂, ̃ and ̣ above the preceding letter, °, ᵇ and ᵇ beside it at the top, ̣ beside it at full height and ̣ below it. (b) A lonsum (final) letter, a cluster joined by apun iyek, the syllable *ai* in everyday and in standard spelling (Section 3), the full stop and digits

which is available only as isolated characters. We compose words from these characters instead, placing each vowel sign according to its zone and syllable.

3 The script and the spelling of i

Meitei Mayek writes 27 letters on a line: consonants, each carrying an inherent vowel a, and the vowel letters 𑜋, 𑜋 and 𑜋 (Fig. 1). Seven vowel signs and the nasal sign nung attach to the preceding letter: three are placed above it, three beside it at the top, inap beside it at full height and unap below it. Eight lonsum letters write the end of a syllable (seven final consonants and i), and apun iyek, written beneath the first of two letters, joins them into a cluster. A sentence ends with cheikhei, the full stop, and there are ten digits. In Unicode a sign follows its letter, which is also the order of writing and of reading from left to right. The 55 classes of TUMMHCD cover all of these characters; the rarely used lum iyek (a heavy tone mark) and the characters of the Meitei Mayek Extensions block are not included.

Two spellings of i.

Both the letter 𑜋 and the lonsum letter 𑜋 write the vowel i. Hijam’s thesis gives the rule [7]: 𑜋 follows a vowel, whereas 𑜋 begins a word or follows a consonant; checked by language experts on about 26,000 words, the rule held for 94.7%, and 𑜋 was about 3.6 times as frequent as 𑜋. We measured the practice of typed text, which records the writer’s choice of letter, in the Meitei Wikipedia [27] and in the Meitei Mayek text of FineWeb-2, both its retained and its filtered-out parts [28]: together 1,454,510 running words and 77,580 distinct words (1,036,283, 52,395 and 365,832 words, respectively, with overlaps). Typed text writes 𑜋 almost exclusively after ̂, ̃ or ̣ (99.4%, 97.1% and 98.8% of all 𑜋 in the three sources), and even in this position it mostly writes 𑜋: in Wikipedia, 𑜋 is 3.5 times as frequent as 𑜋, and the rule of the thesis holds for only 28% to 45% of the i’s in the three sources.

The pages reveal two distinct practices rather than a single noisy one. Of the 4,523 Wikipedia pages with at least five i’s after ̂, ̃ or ̣, 1,615 write 𑜋 for at most 30% of

them and 2,463 for at least 70%, with 445 in between; text outside Wikipedia mostly uses $\mathfrak{S}$ (FineWeb-2: 122 documents against 24). The 605 pages that write $\mathfrak{S}$ in this position at least 90% of the time (55,589 running words) follow a formal standard: under the rule “ $\mathfrak{S}$ after $\mathfrak{e}$, $\mathfrak{o}$ or $\mathfrak{u}$, $\mathfrak{S}$ elsewhere”, they agree for 95.7% of their 11,982 i’s and for 94.8% of their distinct words, close to the 94.7% of the thesis (99.0% without one word, probably inserted by a template, which alone accounts for 398 disagreements). The part of the rule that does not depend on the selection of these pages also holds: they write $\mathfrak{S}$ at the start of a word 1,002 times and $\mathfrak{S}$ 14 times.

In short, the choice between $\mathfrak{S}$ and $\mathfrak{S}$ is a matter of spelling that follows a rule, not a distinction made in handwriting; everyday writing uses $\mathfrak{S}$ throughout, as the writer of our real set confirms from their own practice. Accordingly, we transcribe all text in everyday spelling: the two TUMMHCD classes become a single letter (54 characters in all), every i is written $\mathfrak{S}$, and the standard spelling can be rendered from a reading by the rule. Under this convention, the ensemble of [5] already reaches 98.73% on the TUMMHCD test set.

4 Data

TUMMHCD.

The Tezpur University Meitei Mayek Handwritten Character Database [2] contains 85,124 images, 72,330 for training and 12,794 for testing. We audited it for the requirements of a word synthesiser. The database carries no writer information: there are no folders below the class level and no side files, the file names consist of a class number and a running index, neighbouring files and files with the same number in different classes are unrelated in style, and the 682 distinct file times mark scanning batches rather than writers. All images have the same size, 24×24 pixels, with the ink scaled to fill the frame, so that a character’s size, aspect ratio and position within the word are lost. There are 1,741 groups of pixel-identical images containing 3,512 images, almost all of them pairs of duplicated files; 469 test images (3.7%) have an identical training image, 456 of them with the same label, and 24 groups carry conflicting labels, none of them involving $\mathfrak{S}$ and $\mathfrak{S}$. We use the split of [5], which holds out 15% of each training class for validation: training words are drawn from 61,475 training images, validation words from 10,820 validation images and test words from 12,325 test images, excluding the test images that have a training twin and every image in a group with conflicting labels.

Text.

Native Unicode Meitei Mayek text is scarce. The Meitei Wikipedia (CC BY-SA 4.0) and the Meitei Mayek documents of FineWeb-2 (ODC-By) provide 77,580 distinct words; much Meitei Mayek text on the web is encoded in non-Unicode fonts [29]. After every word has been rewritten in everyday spelling, which merges the two spellings of a word, and strings that cannot be written words have been removed (characters outside the alphabet, a sign at the start, apun not between two consonants, or more than 24 characters; 404 words), 76,066 words remain. A hash of each word assigns it

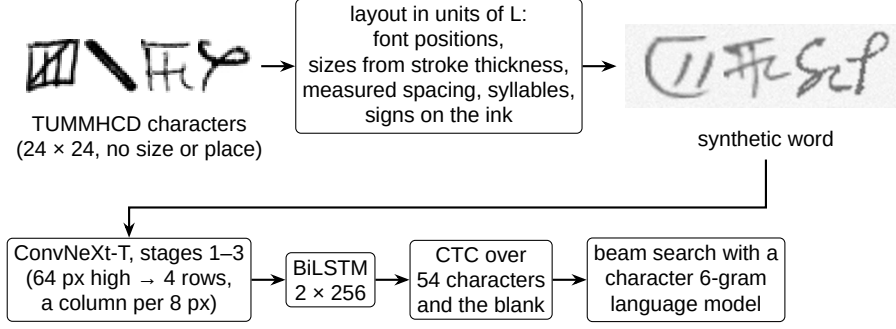


Fig. 2 Training data and recogniser. Words are composed from TUMMHCD character images on a layout that restores what the images lost and are rendered during training; the recogniser reads the whole word image without segmenting it into characters. Character images from TUMMHCD [2]

to training (68,450 words), validation (3,855) or test (3,761), so that no word appears in two splits, whatever its source.

Language model.

A character n -gram model of words with interpolated Kneser–Ney smoothing [30, 31] (discount 0.75) is trained on the training words, numbers and full stops, with the end of a word as a symbol. Its order (3 to 7) and the weighting of the words (each distinct word once, or by the square root or the whole of its count) are chosen by perplexity on the validation words: order 6 with each word counted once gives 7.61 per character (order 5: 7.75; order 7: 7.75; square-root weighting at best 8.15), with 454,480 n -grams. The test words are not among its training words, so the model helps through the syllables, endings and letter sequences of the language rather than by memorising words.

5 Synthetic word generation

A synthetic word is generated as text and then composed from TUMMHCD character images on a layout measured in units of L , the height of a letter (Fig. 2); Fig. 3 shows examples. Composing a word takes about 10 ms on one core of the central processing unit (CPU), so training words are rendered on the fly during training.

Placement from a font.

Because TUMMHCD’s images carry no position, the size and placement of every character are taken from a font, Noto Sans Meetei Mayek [26], shaped with HarfBuzz [32] both alone and after each letter. Letters, lonsum letters and digits stand on the baseline one after another. A sign is placed relative to the pen position after the preceding character, as the font places it, which also covers two signs on one letter (𑜀𑜂𑜆, 𑜀𑜂𑜆); apun lies beneath the whole preceding letter. With all random variation switched off, the synthesiser reproduces the font’s rendering of every sign and combination.

Handwritten sizes from stroke thickness.

Handwriting need not preserve the proportions of print, and TUMMHCD's images, scaled to fill their frame, appear to have lost them. They nevertheless leave a trace: a pen of width p becomes $24p/w$ pixels across a vertical stroke of a character w wide, and $24p/h$ pixels across a horizontal stroke of a character h high. Assuming a single pen width for the whole dataset, the median stroke thicknesses of a class recover its width and height relative to a letter. On the font's own characters, stretched in the same way, the method recovers heights to within a few per cent (median ratio 0.997, 10th to 90th percentile 0.97 to 1.01) and widths to within about 15% (0.94, 0.83 to 1.07). On TUMMHCD, letters are written at about their printed proportions (median width 0.95 and height 1.02 times the font's), whereas the signs are written larger: in height, 2.45 times the font's for $\underline{\circ}$, 1.59 for $\overset{\circ}{\circ}$, 1.49 for $\overset{\circ}{\circ}$, 1.35 for $\circ^\circ$, 1.34 for $\overset{\circ}{\circ}$ and 1.25 for $\circ^\circ$, and $\circ^\circ$ is 1.50 times as wide. The pen is about 0.084 L wide. The synthesiser combines these sizes with the font's positions.

Style and pen.

Without writer identities, a word cannot take all of its characters from one writer. Instead, each image is described by four style features, standardised within its class: slant, stroke width in the frame, ink fraction and ink darkness. A word adopts the style of a random image, and each of its characters is chosen among the 16 images of its class closest to that style. Each character image is resized into its box, and its strokes are then redrawn at the word's pen width (0.06 to 0.14 L) from the signed distance to their edge, since resizing a 24-pixel image scales its strokes with the box. The ink of an image is taken as what its Otsu threshold [33] classifies as ink, so that pale scans keep their strokes. Faint strokes that reach dark ones are preserved, and the centre of a stroke remains at least 130 grey levels darker than the paper after blurring (0.6% of words have strokes less than 100 grey levels below the paper, against 9.7% without this floor).

Spacing and syllables.

In six samples of published Meitei Mayek handwriting (Figs. 3 and 4 of [34] and the four photographs of Fig. 11 of [3]), measured with the same tool, 33.5% of neighbouring letters touch and the others are 0.097 L apart (medians over the samples). A synthetic word joins a character to the preceding one (with touching ink) with a per-word probability of 2% to 60%, and otherwise adds -0.10 to 0.05 L to the font's side bearings; as a result, 32.4% of neighbouring letters touch, with visible gaps of 0.059 L. A syllable, that is, a letter or apun cluster with its vowel sign and its final nung or lonsum letter, is kept together: no gap inside it is wider than the gaps around it, and in one word in five the syllables stand visibly apart.

Signs on the ink.

A sign is placed according to its ink rather than its bounding box. A sign beside its letter is never nearer to the next letter than to its own, is never touched by the next letter and never overlaps its own letter; signs above and below keep their printed

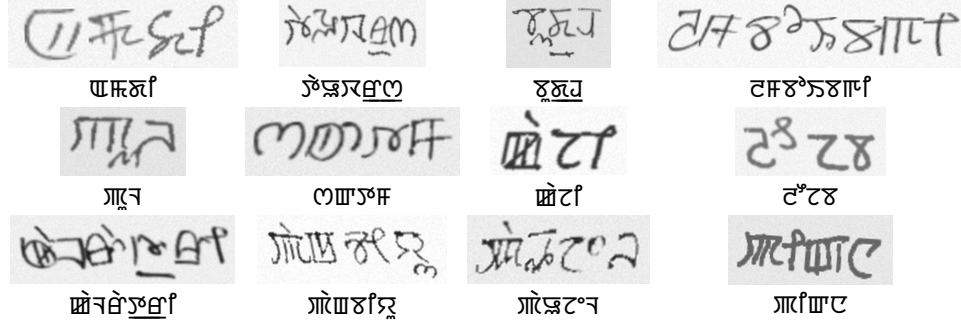


Fig. 3 Synthetic words composed from TUMMHCD validation characters, with their text in print beneath. Character images from TUMMHCD [2]

distance from the letter’s ink and never sit in a hollow of it. On the training characters, with 300 words for each sign, no sign beside its letter is nearer to the next letter (0%) or touched by it (0%), and signs above and below lie at a median distance of 0.085 to 0.10 L from their letter.

Word selection.

Lexicon words are drawn with probability proportional to the square root of their count; 3% of the words are numbers and 2% end with a full stop. Letters that make up less than 0.5% of the characters receive a tenth of the draws, so that 2, for example, rises from 0.017% to 0.23% of the characters drawn. To cover short and long words and every kind of syllable (a letter alone, with a vowel sign, with a vowel sign and a final, or with a final), 15% of the words are composed of one to six real syllables. In the second round of training (Section 8), 10% of the words are lexicon words in which every letter, lonsum letter and vowel sign is replaced by a random one of its kind, so that every letter is seen in every context.

Fixed sets.

A synthetic validation set and a synthetic test set of 5,000 words each are rendered once from the validation and test lexicons and character images, with fixed seeds.

6 Recognition model

Input.

A word image is contrast-normalised by mapping the paper to its 90th percentile of grey level and the ink to its 1st, so that pale ink on grey paper resembles dark ink on white; it is then scaled to a height of 64 pixels with its proportions preserved (at most 1,024 pixels wide), with ink as 1 and paper as 0.

Network.

The encoder consists of the stem and the first three stages of ConvNeXt-T [35], with the downsampling of the third stage modified to halve the height only, so that a word

is represented by 4 rows of 384 channels, with one column for every 8 pixels of width. Each column, with its four rows taken together so that a sign keeps its height above or below its letter, is projected to 512 values and read by a two-layer bidirectional LSTM [36] with 256 units per direction; a linear layer then gives, for each column, the probabilities of the 54 characters and the CTC blank [37]. The fourth stage of ConvNeXt-T, with its stride of 32 pixels, is too coarse for words. The network has 16.3 million parameters, 12.3 million of them in the encoder. One column per 8 pixels leaves every synthetic word more columns than CTC requires, including words with signs above and below.

Initialisation.

The encoder of the main model is initialised from the ConvNeXt-T of [5], trained on the isolated characters of TUMMHCD (the member without size features). We compare it with ConvNeXt-T initialised from ImageNet [38, 39] and with a small CNN trained from scratch, the CRNN of [13], which is the standard baseline for Indic handwritten words [14].

Training.

Each run takes 60,000 steps of 64 words rendered on the fly (3.84 million words), with AdamW [40], a learning rate of 3×10^{-4} , 2,000 warm-up steps and a cosine decay to 1%, a weight decay of 0.05, gradient clipping at 5, bfloat16 precision in the encoder, and an exponential moving average of the weights (0.999), which is evaluated and kept. Words are grouped by width, so that each batch needs little padding. On the graphics processing unit (GPU), every word is augmented towards photographed handwriting: rotation of up to 2° , shear of up to 0.2, changes of width and height, a smooth warp, thicker or thinner strokes, blur, weaker ink, uneven paper, a ruled line and noise, but no flips. Every 2,000 steps, the greedy character error rate on the synthetic validation set selects the checkpoint that is kept. A run takes about 2.5–2.6 hours on one A100 GPU and is limited by the rendering of the words on the CPU (the GPU waits for data about 46% of the time).

Decoding.

Greedy decoding takes the most probable class in each column, merges repeated classes and removes blanks. Beam search (16 prefixes) adds to the CTC log probability of each prefix α times the log probability of the language model, including the end of the word, and a bonus β per character [41]; α and β are chosen on the synthetic validation set.

Evaluation metrics.

The character error rate (CER) is the edit distance summed over all words divided by the number of reference characters, and the word error rate (WER) is the proportion of words not read exactly; word accuracy is reported with a Wilson interval [42]. For each confusable pair, we count how often one member is read as the other. Two recognisers evaluated on the same words are compared with an exact McNemar test [43] on the words that only one of them reads correctly.

Implementation.

All training and evaluation were run in Google Colab on one NVIDIA A100 GPU with 80 GB of memory. The implementation uses PyTorch [44] and timm [39], and the complete code is publicly available (see Statements and Declarations). Claude Code (Anthropic) was used to optimise and polish the source code.

7 Baselines

A segment-then-classify system is given the best segmentation possible: each synthetic validation and test word is rendered again from its seed, which yields the bounding box of every character. Each character is then classified by the six-network ensemble of [5], with $\mathfrak{S}$ and $\mathfrak{S}$ merged, in two forms: its original isolated TUMMHCD image, an upper bound that no segmenter could deliver, and the image cut from the word by its box and scaled to 24×24 like a TUMMHCD image, together with any strokes of neighbouring characters that reach into the box. Each form is decoded in three ways: by the most probable class; with the same character language model, its weight chosen on validation; and with perfect zones, following [6], together with the language model. With perfect zones, a character above the line can only be $\mathfrak{c}$, $\mathfrak{c}$ or $\mathfrak{c}$, one below it $\mathfrak{c}$ or apun, one raised beside its letter $\mathfrak{c}$, $\mathfrak{c}$ or $\mathfrak{c}$, and one on the line any other character. Because the released networks of the ensemble were also trained on our validation characters, its validation choices are made with the development networks of [5], which were trained without them (on isolated images they misread 0.97% of the validation characters). With perfect segmentation and perfect zones, this baseline is stronger than any real segment-then-classify system. The small CNN trained from scratch serves as the segmentation-free baseline.

8 Results on synthetic words

All choices were made on the synthetic validation set (33,175 characters in 5,000 words), and the synthetic test set was read once, by every run and by the baseline together.

Two rounds of training.

The first round trained each of the three encoders once. On validation, every run misread about half of the 64 occurrences of $\mathfrak{K}$ (48% to 52%, greedy), nearly always as $\mathfrak{K}$, although the development networks of [5], reading the same character images in isolation, never read $\mathfrak{K}$ as $\mathfrak{K}$. Since $\mathfrak{K}$ accounts for only 0.009% of the characters of the training text, a prior learned from training words, in which $\mathfrak{K}$ is common and $\mathfrak{K}$ rare, overrode the image. The second round therefore replaced the letters of 10% of the training words with random letters of the same kind (Section 5) and trained the main model with two seeds (for the weights and for the training words), and the ImageNet-initialised model and the small CNN once each. The network alone now misreads fewer occurrences of $\mathfrak{K}$ (23% and 30% for the two seeds, greedy), but the language model, which has never seen $\mathfrak{K}$ in these contexts, pulls the readings back

Table 1 Results on the synthetic validation set (5,000 words). Encoders are listed with their initial weights in brackets. Best step: the checkpoint selected by greedy CER. LM: with the language model, at the weight α and bonus per character β chosen on this set.

Encoder (initial weights)	round	best step	greedy		LM		α, β
			CER	WER	CER	WER	
ConvNeXt-T (TUMMHCD), seed 0	2	60,000	0.274%	1.78%	0.253%	1.68%	0.50, 0.0
ConvNeXt-T (TUMMHCD), seed 1	2	52,000	0.298%	1.92%	0.253%	1.64%	0.50, 0.0
ConvNeXt-T (ImageNet)	2	58,000	0.326%	2.10%	0.268%	1.78%	0.50, 0.5
small CNN (CRNN)	2	54,000	0.298%	1.92%	0.259%	1.70%	0.25, 1.5
ConvNeXt-T (TUMMHCD)	1	58,000	0.304%	2.00%	0.277%	1.80%	0.50, 0.0
ConvNeXt-T (ImageNet)	1	60,000	0.289%	1.88%	0.265%	1.76%	0.25, 0.0
small CNN (CRNN)	1	46,000	0.317%	2.08%	0.265%	1.76%	0.50, 0.0

Table 2 Results on the synthetic test set (5,000 words, 33,475 characters, TUMMHCD test characters), read once. LM: with the language model. Below the line: the ensemble of [5] on the same characters with perfect segmentation, applied to the original isolated images (not attainable by any segmenter) or to the images cut from the word, and decoded as stated in each row; its CER and WER are given in the last two columns.

System	round	greedy		LM	
		CER	WER	CER	WER
ConvNeXt-T (TUMMHCD), seed 0	2	0.329%	2.12%	0.236%	1.52%
ConvNeXt-T (TUMMHCD), seed 1	2	0.370%	2.32%	0.251%	1.62%
mean of the two seeds		0.350%	2.22%	0.244%	1.57%
ConvNeXt-T (ImageNet)	2	0.326%	2.04%	0.245%	1.56%
small CNN (CRNN)	2	0.335%	2.10%	0.275%	1.70%
ConvNeXt-T (TUMMHCD)	1	0.317%	2.04%	0.266%	1.74%
ConvNeXt-T (ImageNet)	1	0.329%	2.08%	0.257%	1.60%
small CNN (CRNN)	1	0.341%	2.20%	0.269%	1.74%
<i>The ensemble of [5] with perfect segmentation:</i>					
isolated images, most probable class				1.204%	7.64%
isolated images, language model				0.245%	1.60%
isolated images, perfect zones, language model				0.173%	1.12%
cut from the word, most probable class				13.634%	54.12%
cut from the word, language model				5.894%	26.38%
cut from the word, perfect zones, language model				4.257%	20.98%

(50% with it). Overall, the two rounds do not differ measurably on the synthetic sets (Table 1); the second round constitutes the final recipe.

Whole words against characters.

On the test set, the final recipe misreads 0.24% of the characters and 1.57% of the words with the language model (mean of two seeds; 0.35% and 2.22% greedy), as shown in Table 2. Cut from the words with perfect boxes, the same characters are read far less accurately by the ensemble: 13.63% of the characters are wrong with the most probable class, and 4.26% with perfect zones and the language model, that is, 17 times the recogniser’s character errors and 13 times its word errors. The cut characters include parts of their neighbours and have lost their size: 𐀓 is read as 𐀔, whose isolated image it resembles, and the box of 𐀕, whose first stroke reaches back over its letter, takes in part of that letter. Only the original isolated images with

perfect zones and the language model (0.17%, 1.12%) outperform the recogniser, and no segmenter can provide those.

The confusable pairs.

As isolated images, the ensemble confuses ◌° and ◌ 94 times and ◌ and ◌ 115 times on the test words, whereas the recogniser never swaps them, in any run or decoding: the position of a sign in the word is decisive. 𑜀 and 𑜁 are swapped 6 to 17 times per run and decoding (ensemble: 51) and remain the hardest pair.

Comparison of runs.

No two of the seven runs differ measurably on the test words: over all 21 pairs, the smallest exact p -value is 0.12 for greedy decoding and 0.18 with the language model. The language model corrects 25 to 50 misread words per run and turns 5 to 22 correct words into errors. On synthetic words, the encoders and rounds cannot be ranked; these results are also in-distribution, since the test words come from the same synthesiser and the writers of TUMMHCD cannot be separated. Lexicon words are read with a WER of 1.05%–1.07% (final recipe, with the language model), and words composed of random syllables with 3.9%–4.7%, which shows that the language model favours real words.

9 Real handwritten words

9.1 Collection

Pages for writing are generated from the test lexicon: each word is printed (shaped with HarfBuzz in Noto Sans Meetei Mayek) above an empty box 2.2 cm high and as wide as the word requires, with four corner squares on each A4 page and the layout embedded in the PDF. The words are distinct test-split words of 2 to 14 characters that occur at least twice in the text, drawn in proportion to the square root of their count after every letter and sign has been given eight occurrences where the test words allow; 5% are numbers and 2% end with a full stop. The 500 items fill 29 pages.

The words were written by the author, a native writer of Meitei Mayek, with an Apple Pencil on an iPad, directly in the PDF of the pages. A cutting tool aligns each written page with its template using the corner squares (a projective transformation, so that photographs of paper pages can also be processed), identifies the page by its printed ink, treats as handwriting whatever is darker than the printed page nearby, assigns each piece of ink to the box it overlaps (a stroke crossing a box line stays with its word and is flagged) and cuts each word from its own ink. A trial page of validation words, which was never scored, showed that the tablet’s thin default pen produced strokes of 2.0 px as seen by the recogniser, compared with 4.0 px in the training words, so the writer chose a pen width that matches them. As the recogniser sees them, the strokes of the real words are 4.0 px wide (0.125 of the letters’ band) and ink covers 14.7% of the image, against 4.0 px, 0.121 and 13.5% for synthetic words.

The writer wrote the first 100 items: 90 lexicon words and 10 numbers with 592 characters (52 of the 54 characters occur; ◌ occurs 3 times and 𑜀 once). Every word

Table 3 Results on the 100 real handwritten words, read once. LM: with the language model. Words right: words read correctly with the language model, with the 95% Wilson interval in %.

Encoder (initial weights)	round	greedy		LM		words right
		CER	WER	CER	WER	
ConvNeXt-T (TUMMHCD), seed 0	2	3.04%	17%	2.20%	12%	88 (80–93)
ConvNeXt-T (TUMMHCD), seed 1	2	4.22%	23%	3.72%	20%	80 (71–87)
mean of the two seeds		3.63%	20.0%	2.96%	16.0%	
ConvNeXt-T (ImageNet)	2	3.04%	17%	2.37%	13%	87 (79–92)
small CNN (CRNN)	2	5.24%	27%	2.87%	16%	84 (76–90)
ConvNeXt-T (TUMMHCD)	1	3.89%	23%	2.70%	16%	84 (76–90)
ConvNeXt-T (ImageNet)	1	3.04%	18%	2.37%	14%	86 (78–91)
small CNN (CRNN)	1	4.39%	24%	3.89%	20%	80 (71–87)

was cut whole: 20 were flagged for a stroke crossing a box line, all of them complete, and no ink was left over. One label was corrected after the test: the writer had not written the full stop printed with one word, every run read that word as written, and it is scored as read correctly; no reading changed. The set is text-disjoint from training: neither the recogniser nor the language model has seen its words, and its writer is not among the writers of TUMMHCD. As the handwriting of a single writer on a tablet, the set shows how the recogniser reads one unseen hand in digital ink rather than how reading varies across hands, and at 80 of 100 words read correctly the 95% interval of word accuracy spans 71% to 87%.

9.2 Results

Every run read the words once, greedily and with the language model at the α and β chosen on synthetic validation (Table 3, Fig. 4).

The final recipe misreads 2.96% of the characters and 16.0% of the words with the language model (mean of the two seeds; 3.63% and 20.0% greedy), about 12 times the character errors on the synthetic test set. The two seeds, which agreed on synthetic words, differ here: with the language model, 9 words are read correctly only by the first seed and 1 only by the second ($p = 0.02$; greedy: 10 against 4, $p = 0.18$). Of the 21 pairs of runs, 3 differ at $p < 0.05$ with the language model, about as many as chance would produce among 21 tests: on 100 words, the encoders and rounds cannot be ranked. The language model corrects 3 to 11 misread words per run and introduces an error in only 1 word across all seven runs. In total, 70 words are read correctly by every run and 6 by none.

Several properties carry over from the synthetic words. The confusable pairs are read without a swap (◦ and O 32 of 32, ◌ and 54–55 of 55, and 20 of 20 in every run, although O and 20 are rare in this set), and so are the letters: of the 56 words with neither ◌ nor ◌, each run misreads between 0 and 2.

9.3 Error analysis

Of the 119 character errors of the seven runs (Table 4), 118 (99%) involve the three signs that print places above the letter: ◌ is omitted (45) or read as ◌ (30) or ◌ (6), ◌ is read as ◌ (28), and ◌ as ◌ or ◌ (7, 2). The 44 words containing ◌ or ◌ are misread



Fig. 4 Real handwritten words with their text and the reading of the final recipe (seed 0, with the language model). (a) Read correctly. (b) Misread, each in 𐌹 or 𐌺, which this writer places after the letter at the top rather than above it

Table 4 All character errors of the seven runs on the real words, with the language model.

written	read as	errors
𐌹	(left out)	45
𐌹	𐌹	30
𐌹	𐌹	28
𐌹	𐌹	7
𐌹	𐌹	6
𐌹	𐌹	2
𐌹	(left out)	1
all		119

11 to 19 times per run, and each run misreads 17% to 36% of the 47 occurrences of 𐌹. The only other error, across all seven runs, is an omitted 𐌹.

Inspection of the misread words suggested the cause, and measurements on the ink confirm it (Table 5, Fig. 5). In 59 of the 76 real words with a sign above, or beside the letter at the top (in the others a sign touches its letter, so that its ink cannot be matched to the text), the writer’s 𐌹 is centred 0.18 L after its letter’s right edge, with its top 0.18 L above the letter’s top line and its bottom 0.19 L below it: it stands after the letter and reaches down beside it, where the writer’s 𐌹 (0.28 L after the edge) and

Table 5 Position of $\grave{\circ}$ measured on the ink: median (10th, 90th percentile), in L, the height of the letters’ band. Centre: horizontal distance from the sign’s centre to the rightmost ink before it (its letter), positive to the right. Top, bottom: its highest and lowest ink relative to the top line, positive above. Slant: direction of its main axis (45° falls to the right, 90° is upright). Elong.: spread along that axis over the spread across it (about 1 for a ring). Written: the real words. Print placement and new placement: 1,000 validation words drawn with TUMMHCD validation characters, with the signs above as print places them and as the new placement puts them (Section 9.4). Below the line: the writer’s $\circ^{\flat}$ and $\circ^{\circ}$.

$\grave{\circ}$	n	centre	top	bottom	slant ($^\circ$)	elong.
written	36	0.18 (0.01, 0.27)	0.18 (0.08, 0.30)	-0.19 (-0.24, -0.04)	52 (43, 60)	4.2
print placement	141	-0.29 (-0.47, 0.65)	0.49 (0.03, 0.71)	0.07 (-0.22, 0.25)	53 (44, 68)	5.2
new placement	129	0.20 (0.03, 0.37)	0.15 (-0.03, 0.30)	-0.27 (-0.48, -0.11)	52 (44, 92)	5.4
written $\circ^{\flat}$	13	0.28 (0.14, 0.36)	0.20 (0.05, 0.31)	-0.42 (-0.53, -0.31)	81 (75, 86)	2.0
written $\circ^{\circ}$	23	0.36 (0.30, 0.48)	0.03 (-0.02, 0.09)	-0.41 (-0.53, -0.32)	105 (91, 128)	1.2

$\circ^{\circ}$ (0.36) stand. $\circ^{\flat}$ and $\circ^{\circ}$ are placed there too. The shape of the writer’s $\grave{\circ}$ matches print and TUMMHCD: a straight stroke falling to the right at 52° (of the 191 TUMMHCD validation images of $\grave{\circ}$, drawn at the size the synthesiser gives them, 97% fall to the right, with a median of 51°). Print, and with it every synthetic word, places $\grave{\circ}$ over the right half of the letter and wholly above its top line: centred 0.29 L before the letter’s right edge, with its bottom 0.07 L above the top line. A recogniser that has only seen $\grave{\circ}$ above its letter thus encounters a stroke in the position of $\circ^{\flat}$ and $\circ^{\circ}$, and either reads it as one of them or ignores it. The letters, drawn from TUMMHCD, carry over; the placement of the signs above, taken from print, does not carry over for this writer.

9.4 Placing the signs above as written

We extended the synthesiser so that, in a chosen share of the words, every sign that print places above the letter ($\grave{\circ}$, $\circ^{\flat}$, $\circ^{\circ}$) is written on the line after the preceding character, as $\circ^{\flat}$ is: its left end lies -0.1 to 0.1 L from that character’s right edge and its top 0.06 to 0.3 L above its letter’s top (the writer’s 10th to 90th percentiles are -0.14 to 0.12 and 0.08 to 0.30), and it reaches at most 0.5 L below that top. From there it follows the rules of the signs beside a letter, and the shapes remain those of TUMMHCD. Drawn with TUMMHCD’s validation characters, $\grave{\circ}$ is then centred 0.20 L after its letter, with its bottom 0.27 L below the top line, at 52° (Table 5). A sign above is nearer to the next letter than to its own in 1 of 839 words, and the next letter never touches it. With the share set to zero, every word renders exactly as before, so that the fixed sets remain valid.

Whether training with this placement improves the reading of real handwriting is left to future work. Because the first 100 real words shaped this change, they would then serve as a development set, and the test would require new words, by this writer (the prepared pages hold 400 more items) and by others. A synthetic set of the validation words with every sign above written beside its letter would separate the effect of the placement from that of the writer.

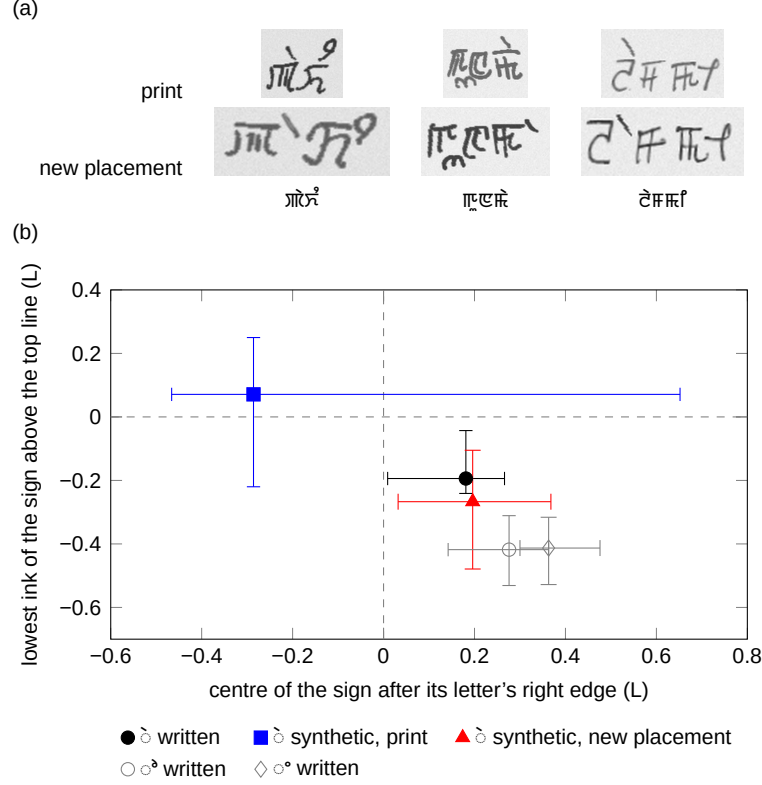


Fig. 5 Placement of the signs above. (a) The same words drawn with TUMMHCD validation characters, with $\mathfrak{e}$ placed as in print (upper row) and by the new placement (lower row). (b) Position of the ink of a sign relative to its letter's top-right corner at (0, 0): median, with bars from the 10th to the 90th percentile (Table 5). The writer's $\mathfrak{e}$ stands where the writer's $\mathfrak{o}^{\mathfrak{a}}$ and $\mathfrak{o}^{\mathfrak{o}}$ stand, not where print, and the synthetic words used for training, place it. Character images from TUMMHCD [2]

10 Browser demonstration

The recogniser also runs in a web page, on the visitor's own device (Fig. 6). The network is exported to the Open Neural Network Exchange (ONNX) format with float16 weights (32.7 MB) and run by ONNX Runtime Web [45]; on all 300 validation words checked, it gives the same greedy reading as PyTorch [44]. The language model is exported in back-off form (669,149 entries, 3.2 MB compressed), and the preprocessing and the beam search are ported to JavaScript; on test inputs, the page produces the same images and readings as the Python code. A visitor writes a word on the page or uploads an image of one, and the page shows the reading with and without the language model, the alternatives found by the beam search and, on request, the reading in standard spelling ($\mathfrak{S}$ after $\mathfrak{e}$, $\mathfrak{o}^{\mathfrak{a}}$ or $\mathfrak{o}^{\mathfrak{o}}$). The page runs the second seed of the final recipe, chosen on synthetic validation before the real test; on the real words it is the weaker of the two seeds (CER 3.72%, WER 20% with the language model).



Fig. 6 The browser demonstration (<https://huggingface.co/spaces/Chingkheinganba/handwritten-meitei-mayek-word-recognition>) reading the author’s name, written in its canvas: the reading with the language model, which the network alone also gives, and the alternatives of the beam search with their probabilities. The example was written by the writer of the real set and is shown for illustration only; it is not part of the evaluation

11 Discussion

Strengths and limits of synthetic training words.

Composed from isolated characters, the synthetic words taught the recogniser the letters, the digits, the signs beside and below a letter, and the confusable pairs well enough to transfer to an unseen writer in digital ink. What did not transfer was a convention of placement taken from print. This indicates where synthetic data is weakest: not in what the character images show, which comes from many writers, but in what the synthesiser decides, which comes from a single source. Measuring real handwriting, as we did for letter spacing and then for the signs above, turns those decisions into data.

Limitations.

The real set comprises one writer, one device and 100 words; the placement of the signs above was measured on this one hand, and whether other writers place them in the same way is not known. The synthetic test results are in-distribution and, on their own, say little about real handwriting. TUMMHCD has no writer identities, so the synthetic words mix writers within a word, matched only in style, and no synthetic split can be writer-disjoint. A comparison with the word-level work of Hijam and Saharia [6] and with Hijam’s thesis [7] on their data is not possible, since those data are not public; instead, our baseline gives their approach perfect segmentation and perfect zones. The language model is built from Wikipedia and web text, whose spelling, vocabulary and loanwords differ from everyday writing.

Future work.

The next steps are training with the new placement of the signs above, tested on new words; more writers, on paper and on tablets; the remaining prepared pages, which would also provide a real development set; a check of the style matching; and generated handwriting in different writers’ styles, judged by whether it improves the reading of real words.

12 Conclusion

We have presented the first segmentation-free recogniser of handwritten Meitei Mayek words. Trained only on words composed from isolated characters, it reads whole handwritten words: on synthetic test words it makes a small fraction of the errors of the best isolated-character ensemble, even when that ensemble is given perfect segmentation and zones, and it resolves the confusable pairs from their position in the word. On a first real test, one writer’s words in digital ink, it reads the letters and digits reliably and misreads almost only the signs that this writer places differently from print, a placement that the synthesiser can now reproduce. Finally, the spelling of *i*, the largest source of error in isolated-character recognition, turns out to be a convention that a recogniser need not resolve from the image.

Statements and Declarations

- **Funding:** The author received no financial support for this work.
- **Competing interests:** The author declares no competing interests.
- **Ethics approval and consent to participate:** Not applicable; the study uses a public dataset of anonymous handwriting samples, and the real word set was written by the author, so no data were collected from other people.
- **Consent for publication:** Not applicable.
- **Data availability:** TUMMHCD [2] is available from its maintainers at <http://agnigarh.tezu.ernet.in/~sarat/resources.html>. The real word set (100 word images, their labels and a manifest) is released under CC BY 4.0 in the code repository (`real_words/`) and archived with the code on Zenodo, <https://doi.org/10.5281/zenodo.23002147>. The Meitei Wikipedia dump [27] (CC BY-SA 4.0)

and FineWeb-2 [28] (ODC-By) are public, and the word lists are rebuilt from them by the released scripts. The numbers in the tables and figures of this paper are in the results files written by the code (`results/` in the code repository).

- **Code availability:** The code (synthesiser, recogniser, training, evaluation, baseline, writing pages and cutting tool, demonstration and tests) is available at <https://github.com/chingkheinganba231005/meitei-mayek-word-recognition> under the MIT licence and archived on Zenodo, <https://doi.org/10.5281/zenodo.23002147> [46]. The trained recogniser and the language model (CC BY-SA 4.0) are available at <https://huggingface.co/Chingkheinganba/handwritten-meitei-mayek-word-recognition>, and the interactive demonstration at <https://huggingface.co/spaces/Chingkheinganba/handwritten-meitei-mayek-word-recognition>. Seeds: 42 for the split of the characters; 0 for the weights and 1000 for the training words of every run, except the second run of the final recipe (1 and 1001).
- **Author contributions:** C.R. is the sole author and conceived and designed the study, wrote the real word set, ran the experiments and prepared the manuscript, with the assistance stated in Section 6.

References

- [1] The Unicode Consortium: The Unicode Standard, Version 16.0.0. The Unicode Consortium, Mountain View, CA (2024). Meitei Mayek block U+ABC0–U+ABFF
- [2] Hijam, D., Saharia, S.: On developing complete character set Meitei Mayek handwritten character database. *The Visual Computer* **38**, 525–539 (2022) <https://doi.org/10.1007/s00371-020-02032-y>
- [3] Inunganbi, S., Choudhary, P., Manglem, K.: Meitei Mayek handwritten dataset: compilation, segmentation, and character recognition. *The Visual Computer* **37**, 291–305 (2021) <https://doi.org/10.1007/s00371-020-01799-4>
- [4] Nongmeikapam, K., Wahengbam, K., Meitei, O.N., Tuithung, T.: Handwritten Manipuri Meitei-Mayek classification using convolutional neural network. *ACM Transactions on Asian and Low-Resource Language Information Processing* **18**(4), 35 (2019) <https://doi.org/10.1145/3309497>
- [5] Rajkumar, C.: Handwritten Meitei Mayek Recognition: code, trained networks and results of the character-level study, version 1.0.0. GitHub, <https://github.com/chingkheinganba231005/Handwritten-Meitei-Mayek-Recognition>; archived on Zenodo, <https://doi.org/10.5281/zenodo.22932285>. Accessed 27 September 2026 (2026)
- [6] Hijam, D., Saharia, S.: Zone and rule assisted recognition of Meitei-Mayek handwritten characters. *Evolutionary Intelligence* **17**, 2963–2980 (2024) <https://doi.org/10.1007/s12065-024-00920-z>

- [7] Hijam, D.: Convolutional neural network assisted off-line handwritten character recognition of Meitei Mayek. PhD thesis, Department of Computer Science and Engineering, Tezpur University, Tezpur, India (2024). <http://agnee.tezu.ernet.in:8082/jspui/handle/1994/1707>. Accessed 27 September 2026
- [8] Mondal, A., Jawahar, C.V.: Unconstrained camera captured Indic offline handwritten dataset. In: Pattern Recognition (ICPR 2024). Lecture Notes in Computer Science. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-78495-8_21
- [9] MWire Labs: NE-OCR: unified optical character recognition for 10 languages of Northeast India. Research Square preprint, <https://www.researchsquare.com/article/rs-9167777/v1>. Accessed 27 September 2026 (2026)
- [10] Naosekham, V., Islam, M., Chourasia, A., Sahu, N.: EMBiL: an English-Manipuri bi-lingual benchmark for scene text detection and language identification. In: Computer Analysis of Images and Patterns (CAIP 2023). Lecture Notes in Computer Science. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-44237-7_7
- [11] Hijam, D., Saharia, S.: A multilevel recognition of Meitei Mayek handwritten characters using fusion of features strategy. *The Visual Computer* **40**, 211–225 (2024) <https://doi.org/10.1007/s00371-023-02776-3>
- [12] Roy, P.P., Bhunia, A.K., Das, A., Dey, P., Pal, U.: HMM-based Indic handwritten word recognition using zone segmentation. *Pattern Recognition* **60**, 1057–1075 (2016) <https://doi.org/10.1016/j.patcog.2016.04.012>
- [13] 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) <https://doi.org/10.1109/TPAMI.2016.2646371>
- [14] Gongidi, S., Jawahar, C.V.: iiit-indic-hw-words: a dataset for Indic handwritten text recognition. In: Document Analysis and Recognition – ICDAR 2021. Lecture Notes in Computer Science. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86337-1_30
- [15] Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. arXiv:1406.2227 (2014). <https://doi.org/10.48550/arXiv.1406.2227>
- [16] Krishnan, P., Jawahar, C.V.: Generating synthetic data for text recognition. arXiv:1608.04224 (2016). <https://doi.org/10.48550/arXiv.1608.04224>
- [17] Wigington, C., Stewart, S., Davis, B.L., Barrett, B., Price, B.L., Cohen, S.: Data augmentation for recognition of handwritten words and lines using a CNN-LSTM

- network. In: 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp. 639–645 (2017). <https://doi.org/10.1109/ICDAR.2017.110>
- [18] Roy, P.P., Mohta, A., Chaudhuri, B.B.: Synthetic data generation for Indic handwritten text recognition. arXiv:1804.06254 (2018). <https://doi.org/10.48550/arXiv.1804.06254>
 - [19] Kang, L., Riba, P., Wang, Y., Rusiñol, M., Fornés, A., Villegas, M.: GANwriting: content-conditioned generation of styled handwritten word images. In: Computer Vision – ECCV 2020. Lecture Notes in Computer Science, pp. 273–289. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58592-1_17
 - [20] Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., Litman, R.: ScrabbleGAN: semi-supervised varying length handwritten text generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4324–4333 (2020). <https://doi.org/10.1109/CVPR42600.2020.00438>
 - [21] Bhunia, A.K., Khan, S., Cholakkal, H., Anwer, R.M., Khan, F.S., Shah, M.: Handwriting transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1086–1094 (2021). <https://doi.org/10.1109/ICCV48922.2021.00112>
 - [22] Nikolaidou, K., Retsinas, G., Christlein, V., Seuret, M., Sfikas, G., Barney Smith, E., Mokayed, H., Liwicki, M.: WordStylist: styled verbatim handwritten text generation with latent diffusion models. In: Document Analysis and Recognition – ICDAR 2023. Lecture Notes in Computer Science, vol. 14188, pp. 384–401. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-41679-8_22
 - [23] Nikolaidou, K., Retsinas, G., Sfikas, G., Liwicki, M.: DiffusionPen: towards controlling the style of handwritten text generation. In: Computer Vision – ECCV 2024. Lecture Notes in Computer Science, pp. 417–434. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-73013-9_24
 - [24] Dai, G., Zhang, Y., Ke, Q., Guo, Q., Huang, S.: One-DM: one-shot diffusion mimicker for handwritten text generation. In: Computer Vision – ECCV 2024. Lecture Notes in Computer Science. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-73636-0_24
 - [25] Wolf, F., Fink, G.A.: Self-training of handwritten word recognition for synthetic-to-real adaptation. In: 26th International Conference on Pattern Recognition (ICPR), pp. 3885–3892 (2022). <https://doi.org/10.1109/ICPR56361.2022.9956168>
 - [26] The Noto Project Authors: Noto Sans Meetei Mayek, version 2.002. <https://github.com/notofonts/meetei-mayek>; SIL Open Font Licence 1.1. Accessed 27

- [27] Wikimedia Foundation: Meitei Wikipedia database dump (mniwiki), pages and articles. <https://dumps.wikimedia.org/mniwiki/>; CC BY-SA 4.0. Accessed 27 September 2026 (2026)
- [28] Penedo, G., Kydlíček, H., Sabolčec, V., Messmer, B., Foroutan, N., Kargaran, A.H., Raffel, C., Jaggi, M., Von Werra, L., Wolf, T.: FineWeb2: one pipeline to scale them all – adapting pre-training data processing to every language. arXiv:2506.20920; data: <https://huggingface.co/datasets/HuggingFaceFW/fineweb-2>. Accessed 27 September 2026 (2025). <https://doi.org/10.48550/arXiv.2506.20920>
- [29] Kudugunta, S., Caswell, I., Zhang, B., Garcia, X., *et al.*: MADLAD-400: a multilingual and document-level large audited dataset. In: Advances in Neural Information Processing Systems 36, Datasets and Benchmarks Track (2023)
- [30] Kneser, R., Ney, H.: Improved backing-off for m-gram language modeling. In: International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1, pp. 181–184 (1995). <https://doi.org/10.1109/ICASSP.1995.479394>
- [31] Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. Computer Speech & Language **13**(4), 359–394 (1999) <https://doi.org/10.1006/csla.1999.0128>
- [32] HarfBuzz contributors: HarfBuzz text shaping engine. <https://harfbuzz.github.io/>. Accessed 27 September 2026 (2026)
- [33] Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics **9**(1), 62–66 (1979) <https://doi.org/10.1109/TSMC.1979.4310076>
- [34] Laishram, R., Singh, P.B., Singh, T.S.D., Anilkumar, S., Singh, A.U.: A neural network based handwritten Meitei Mayek alphabet optical character recognition system. In: 2014 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp. 1–5 (2014). <https://doi.org/10.1109/ICCIC.2014.7238510>
- [35] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., Xie, S.: A ConvNet for the 2020s. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11976–11986 (2022). <https://doi.org/10.1109/CVPR52688.2022.01167>
- [36] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997) <https://doi.org/10.1162/neco.1997.9.8.1735>
- [37] Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal

- classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning (ICML), pp. 369–376 (2006). <https://doi.org/10.1145/1143844.1143891>
- [38] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248–255 (2009). <https://doi.org/10.1109/CVPR.2009.5206848>
 - [39] Wightman, R.: PyTorch Image Models. <https://github.com/huggingface/pytorch-image-models>. Accessed 27 September 2026 (2019). <https://doi.org/10.5281/zenodo.4414861>
 - [40] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2019)
 - [41] Hannun, A.Y., Maas, A.L., Jurafsky, D., Ng, A.Y.: First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs. arXiv:1408.2873 (2014). <https://doi.org/10.48550/arXiv.1408.2873>
 - [42] Wilson, E.B.: Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association **22**(158), 209–212 (1927) <https://doi.org/10.1080/01621459.1927.10502953>
 - [43] McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika **12**(2), 153–157 (1947) <https://doi.org/10.1007/BF02295996>
 - [44] Paszke, A., *et al.*: PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32 (NeurIPS) (2019)
 - [45] ONNX Runtime developers: ONNX Runtime. <https://onnxruntime.ai/>. Accessed 27 September 2026 (2021)
 - [46] Rajkumar, C.: Handwritten Meitei Mayek word recognition: code, synthesiser, results and demonstration. <https://github.com/chingkheinganba231005/meitei-mayek-word-recognition>; models and demonstration: <https://huggingface.co/Chingkheinganba/handwritten-meitei-mayek-word-recognition>. Accessed 27 September 2026 (2026). <https://doi.org/10.5281/zenodo.23002147>