Techno-Science Research Journal V14 (Special Issue) (2026) P 66-77

Content list available at ITC

Techno-Science Research Journal

Journal Homepage: http://techno-srj.itc.edu.kh/

Techno-Science Research Journal

KVerifyID: A Hybrid Multimodal Approach for Khmer Online Writer Verification

Ngin Kimlong[1, 4*], Valy Dona[1], Phauk Sokkhey[3], Pin Vannaro[3], Yin Bamnang[4], Ny Seangleng[4], Pen Saron[4]

[1] *Mechatronics and Information Technology, Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh, Cambodia*

[2] *Department of Applied mathematics and statistics (AMS), Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh, Cambodia*

[3]*University of Heng Samrin Thbongkhmum, Cambodia*

[4]*Institute of Information Technology, University of Heng Samrin Thbongkhmum, Cambodia*

Received: 09 February 2026; Revised: 26 February 2026; Accepted: 01 March 2026; Available online: 31 July 2026

Abstract: *Online writer verification with dynamic handwriting signals is still difficult, and it has been especially under-studied for complex Southeast Asian scripts like Khmer. This work tackles online, text-independent, word-level Khmer writer verification as a pairwise decision problem: given two handwritten word samples, decide whether they were written by the same person. We introduce **KVerifyID**, a hybrid dual-stream Siamese network that learns from both (i) a grayscale image rendering of each word and (ii) its pen-trajectory sequence $(x, y, p)$with explicit pen-state encoding. The resulting modality embeddings are fused into a compact 128-dimensional writer representation, and verification is performed via cosine similarity, using thresholds selected on validation and then fixed for testing. On a Khmer online handwriting dataset collected from 298 writers (4,878 word instances) with strict writer-disjoint splits, the model generalizes strongly, achieving 99.50% training accuracy and 99.74% test accuracy, with a low verification error of 0.32% validation EER (equal error rate). At the validation equal-error operating point, the errors are FAR (false acceptance rate) = 0.30% and FRR (false rejection rate) = 0.19%. Overall, the results show that jointly leveraging spatial word appearance and online stroke dynamics enables robust Khmer writer verification, making it promising for digital authentication and forensic screening.*

Keywords: KVerifyID; writer verification; online handwriting; Khmer script; Siamese neural network; multimodal fusion; pen trajectory; error rate;

1. INTRODUCTION

1.1 Background

Handwriting biometrics captures distinctive motor behaviors that may reflect aspects of a writer's identity and, in some contexts, psychological state [1],[2]. In security and forensic applications, a central challenge is to reliably distinguish individuals based on their handwriting characteristics [3]. This task is challenging because handwriting exhibits substantial variation both between writers and within the same writer across sessions, devices, and writing conditions [4][5].

In the literature, a clear distinction is typically made between writer identification and writer verification. Writer identification aims to determine which writer in a candidate set most likely produced a questioned sample (a 1:N problem), whereas writer verification decides whether two given samples were written by the same person or by different

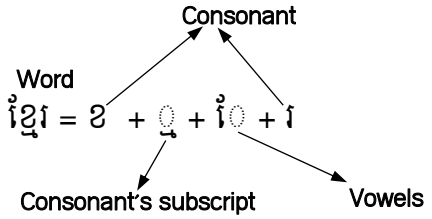

Fig. 1 Illustration of Khmer character composition: a word is formed by combining a base consonant with dependent vowels, diacritics, and subscript consonants, creating a compact multi-layer 2D structure that increases script complexity for handwriting analysis.

**Corresponding author: Ngin Kimlong*
E-mail: nginkimlong@gmail.com

people (a 1:1 decision). Another important distinction is between offline and online handwriting. Offline systems operate on static images of ink traces and therefore have limited access to dynamic writing information. Online systems, in contrast, record time-stamped pen trajectories, including (x, y) −coordinates and pen-state information, and may additionally capture pressure and speed enabling richer representations but also requiring robust modeling of temporal dynamics and coping with sensor noise [6], [7].

In this study, we address online, text-independent, word-level writer verification for Khmer script. Verification is formulated as a binary decision on pairs of word instances: given two samples, a Siamese similarity-learning model predicts whether they originate from the same writer or from different writers. This setting emphasizes generalization under intra-writer variation (e.g., session or device changes) and inter-writer similarity, which can be amplified by Khmer's orthographic structure and writing conventions.

Handwriting biometrics has long been studied for security and forensic applications. For example, Chaski [8] introduced a computational stylometric technique for authorship attribution in digital forensic investigations, and Faundez-Zanuy et al. [6] surveyed handwriting biometrics in e-security and e-health applications, emphasizing both practical deployments and open research challenges. Earlier approaches often relied on handcrafted descriptors and physical characteristics such as writing medium, character size and style, and spacing between characters and words [9]. In online settings, trajectory-based methods that analyze sequences of spatial coordinates (x, y) and pen-state indicators (p) have shown strong performance, and dedicated systems have been developed for major scripts such as English [11], and Bengali [12]. However, comparable deep learning–based online writer verification systems for Khmer script remain limited, motivating the present study.

1.2 Challenges

Khmer is an abugida writing system of the Mon–Khmer branch of the Austroasiatic language family and is primarily used to write the Khmer language. Its structural richness and large symbol inventory pose substantial challenges for computational handwriting systems [12]. Contemporary Khmer includes 33 basic consonants [13], 17 dependent vowels [14], and approximately 14 diacritic marks [15] that may appear above, below, before, or after the base character. These components combine into compact consonant–vowel–diacritic clusters and may involve stacked subscripts and complex two-dimensional layouts (Fig. 1), resulting in higher visual and structural complexity than many alphabetic scripts.

For online writer verification, these script-level challenges are compounded by significant intra-writer variability (e.g., changes across sessions) and inter-writer differences in stroke order, pen-lifts, pen trajectories, and stylistic realizations of the same character cluster. Progress is further constrained by the limited availability of large, publicly accessible online Khmer handwriting datasets and by the lack of models explicitly tailored to Khmer's orthographic structure. Consequently, Khmer writer verification, particularly in the online, text-independent, word-level setting studied in this work benefits from hybrid architectures that can jointly capture fine-grained spatial word-shape cues and temporal stroke dynamics, while remaining robust under low-resource data conditions.

1.3 Contributions

To bridge this gap, we introduce KVerifyID, a deep-learning framework for online Khmer handwriting writer verification. Our main contributions are: Hybrid Siamese architecture:

- **Hybrid dual-stream Siamese verification model:** We propose a Siamese architecture that fuses a word-image stream (rendered grayscale word shape) with a trajectory stream (x, y, p) capturing pen-state dynamics, producing a compact 128-D writer embedding for similarity-based verification.
- **Writer-disjoint dataset and protocol:** We curate a Khmer online handwriting dataset and evaluate verification under strict writer-disjoint train/validation/test splits, preventing identity leakage and measuring generalization to unseen writers.
- **Khmer-aware preprocessing:** We standardize coordinate sequences through normalization and scaling to $[0, 1]$ and include explicit pen-state encoding to preserve multi-stroke structure.
- **Strong performance with low error:** KVerifyID achieves 99.50% training accuracy and 99.74% test accuracy, with Val EER = 0.32% and low operating errors (FAR = 0.30%, FRR = 0.19%) at the validation-selected equal-error threshold.

2. RELATED WORKS

Handwriting-based writer verification is a long-standing topic in pattern recognition, studied in both offline and online settings for applications in forensic document analysis, identity verification, and digital security. As a behavioral biometric trait, handwriting can encode distinctive motor patterns of an individual and may also correlate with aspects of psychological state [1]. Early work in this area was largely offline, where models analyze static images of handwritten text. With the widespread adoption of digital pens and touch devices, online handwriting analysis has gained increased attention because it provides temporal and dynamic signals such as stroke order, coordinate trajectories (x, y), pen pressure, and writing speed [16],[17]. These online signals often contain discriminative cues that are not accessible from offline

images alone, and they can improve robustness for verification tasks.

The rapid progress of deep learning has further advanced writer verification by enabling end-to-end representation learning from raw handwriting inputs, reducing reliance on handcrafted features. Earlier approaches commonly used statistical and classical machine-learning models such as Hidden Markov Models (HMMs) and Support Vector Machines (SVMs), whereas more recent systems leverage deep architectures that learn features directly from images and/or trajectory sequences [18]. A key milestone was introduced by Bromley et al. [19], who proposed the Siamese Neural Network (SNN) framework for verification. Siamese models have become widely used in handwriting biometrics because they learn an embedding space where samples from the same writer are closer than samples from different writers, and verification can be performed via a similarity score (e.g., cosine similarity or Euclidean distance) followed by thresholding. Recent studies, such as DeepWriterID [20], demonstrate strong performance using deep Siamese pipelines that combine complementary cues from spatial appearance and temporal dynamics. Many deep verification pipelines also adopt normalization and coordinate scaling to standardize inputs and improve training stability [21].

Beyond general frameworks, script-specific systems for writer identification and handwriting recognition have been developed for structurally complex scripts such as Arabic, Bengali, Devanagari, and Chinese, as well as for English [22].

These studies emphasize that robust performance often depends on capturing script-dependent stroke structure, character composition, and layout patterns, especially when content varies (text-independent protocols) and writing conditions change. However, Khmer remains comparatively under-explored in online handwriting biometrics despite its broad usage in Cambodia and Khmer-speaking communities abroad [23],[24]. In particular, prior work has not systematically investigated deep online writer verification for Khmer under rigorous writer-disjoint evaluation protocols. This gap motivates the present study, which introduces an online Khmer handwriting dataset and proposes a hybrid Siamese verification approach designed to integrate temporal stroke dynamics with the spatial structure of rendered Khmer words.

3. METHODOLOGY

3.1 Dataset and pair construction

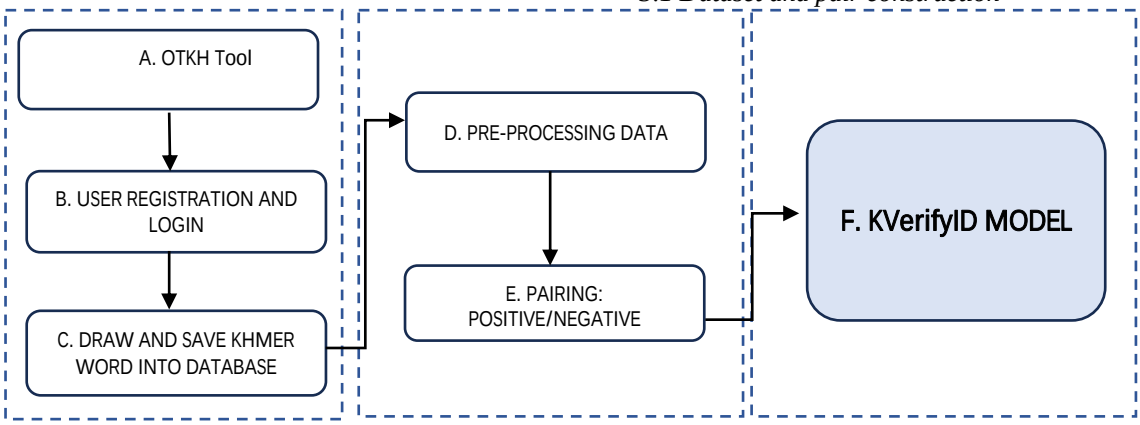

Fig. 2 Overall pipeline of the KVerifyID study: handwriting data are collected using the OTKH tool (registration/login and word capture), stored in a database, then preprocessed and paired into genuine/impostor word pairs. The paired samples are used to train the KVerifyID Siamese model, which outputs a cosine-similarity score for writer verification (same-writer vs. different-writer).

We formulate online Khmer writer verification as a binary verification task on pairs of word instances. Each pair is labeled as genuine (same writer) or impostor (different writers) and is used to train and evaluate the Siamese verification model.

3.1.1 Data collection and representation

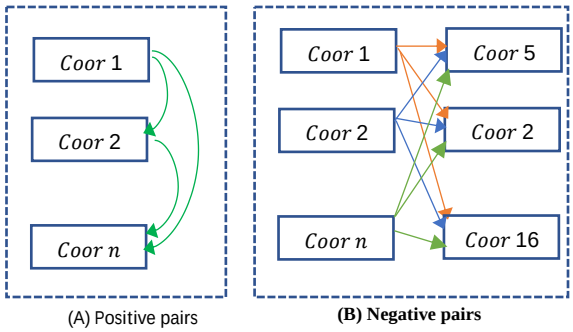

Fig. 3 Pair construction in the KVerifyID dataset. (A) **Genuine pairs: each sample is paired with other samples from the same writer (excluding self-pairs) to capture intra-writer variation.** (B) **Impostor pairs: each sample is paired with samples from different writers to create diverse inter-writer comparisons for Siamese training.**

a) The KVerifyID dataset was collected using the Online Tools for Khmer Handwriting (OTKH) platform (Appendix A), which records online handwriting from

multiple Khmer writers. Each sample corresponds to a single handwritten word and includes: a grayscale rendered image of the word trace, and

b) An online stroke trajectory represented as a sequence of (x, y, p) points, where p denotes the pen-state, together with the associated writer identity label.

In total, the dataset contains 4,878 instances of handwritten words from 298 unique writers, with an average of approximately 16 samples per writer.

3.1.2 Writer-disjoint split protocol

To ensure rigorous evaluation, we use a writer-disjoint protocol: writers are partitioned into non-overlapping training, validation, and test splits (70/10/20) using a fixed random seed. Pair generation is performed only within each split, ensuring that no writer appears in more than one split and that performance reflects generalization to unseen writers.

Within each split, we generate positive (genuine) and negative (impostor) pairs as follows:

- **Positive pairs (genuine):** For each writer, we form all unordered pairs (combinations) among that writer's word instances within the same split, excluding self-pairs (Fig. 3(A)). These genuine pairs capture intra-writer variability due to lexical differences, writing speed, and writing conditions.
- **Negative pairs (impostor):** For each word instance, we randomly sample 16 impostor matches by pairing it with instances from different writers within the same split (Fig. 3(B)). This sampling strategy produces diverse inter-writer comparisons while controlling the class balance for Siamese training.

After generating all available pairs, we obtained 241,355 total pairs. We then enforced the writer-disjoint protocol and retained the final set of 176,347 pairs, distributed as follows:

- **Training:** 137,269 pairs
- **Validation:** 11,503 pairs
- **Test:** 27,575 pairs

This structured writer-disjoint pairing procedure provides a diverse set of genuine and impostor examples and prevents identity leakage, ensuring that the reported results reflect true generalization rather than memorization of writers.

3.2 Threshold selection and test evaluation

Given a handwriting pair, the Siamese model outputs a similarity score s (cosine similarity). We determine decision thresholds only on validation and then freeze them for test reporting.

Validation-fixed thresholds and test evaluation: On the validation split, we sweep similarity thresholds to locate the equal-error threshold τ_EER where FAR and FRR intersect (with linear interpolation). We also set $\tau_{FAR} = \alpha$ to achieve a target impostor acceptance (e.g., $\alpha = 1\%$) by taking the $(1 - \alpha) - quantile$ of the validation impostor-score distribution. These thresholds are then frozen and applied to the test split to report:

- At τ_{EER}: ERR and FAR/FRR@EER;

At $\tau_{FAR} = 1\%$: Acc/F1/FRR at *FAR=1%*.

3.3 Incorporating pen-state into coordinate data

To enhance the handwriting data with temporal stroke dynamics, a pen-state feature was introduced alongside the (x, y) coordinates, resulting in triplet representations of the form (x_t, y_t, p_t), where t denotes the time step. The pen-state p_t encodes critical writing actions:

$$S = \{(x_t, y_t)\}_{t=1}^{T}\ x_t, y_t \in \mathbb{R} \quad (1)$$

The pen-state variable p_t encodes writing activity, defined as: $p_t \in \{0, 1, 2\}$

- $p_t = 0$ represent pen-down (actively writing)
- $p_t = 1$ represent Pen-up (pen lifted)
- $p_t = 2$ represent end of sequence

Each data point is then extended to:

$$s_t = (x_t, y_t, p_t) \in \mathbb{R}^2 \times \{0, 1, 2\} \quad (2)$$

The full sequence is represented as:

$$S' = \{(x_t, y_t\ p_t)\}_{t=1}^{T} \quad (3)$$

Where,

T is the total number of points in the sequence. This structured representation enables the model to capture both spatial patterns and writing behaviors, which are essential for tasks of writer verification.

3.4 Feature scaling

To reduce variability in stroke size and writing extent across samples, we apply min–max normalization to all coordinate values so that each trajectory is mapped into the $[0, 1]$ range. This scaling improves training stability and convergence by keeping input magnitudes consistent across writers and devices, reducing the chance of unstable gradient updates, and aligning the input scale with common neural network activations. Let x and y denote the horizontal and vertical coordinates of points in a trajectory. For each sample, we compute the minimum and maximum coordinate values and normalize each point as:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4)$$

$$y_{scaled} = \frac{y - y_{min}}{y_{max} - y_{min}} \quad (5)$$

Where, x, y represent the raw coordinate values of each point in the stroke sequence. x_{min}, x_{max} are the minimum and maximum x-values in a given handwriting sample. y_{min}, y_{max} are the corresponding minimum and maximum y-values. x_{scaled}, y_{scaled} are the resulting normalized coordinates within the $[0,1]$ range.

To maintain the structural integrity of stroke boundaries during preprocessing, a sentinel value of -1 was inserted as

a marker to denote breaks between individual strokes. This strategy enables the model to distinguish between continuous and segmented strokes without disrupting the normalized coordinate range.

3.5 Coordinate normalization

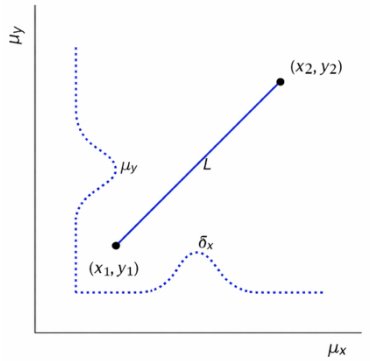

Fig. 4 Coordinate normalization for online handwriting trajectories. For each sample, successive points define line segments *L*, from which length-weighted statistics (μ_x, μ_u) and a scale term δ_x are estimated. All (x, y) points are then translated and scaled to obtain a standardized coordinate space, improving alignment and reducing device- and style-dependent variation.

Another source of variability arises from differences in scale and absolute coordinate values across samples, which may result from diverse input devices, writing areas, or individual handwriting styles. To improve geometric consistency, we normalize the (x, y)-coordinates using a translation-and-scale transformation based on a length-weighted mean and standard deviation computed from stroke segments. As illustrated in Fig. 4, let L denote a straight-line segment connecting two successive points (x_1, y_1) and (x_2, y_2).

The projections of this line onto x-axis and y-axis are defined as:

$$p_x(L) = \int_L x dL = \frac{1}{2} len(L)(x_1 + x_2), \quad (6)$$

$$p_y(L) = \int_L y dL = \frac{1}{2} len(L)(y_1 + y_2), \quad (7)$$

Where,

$$len(L) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

denotes the length of L. Using these projections, the length-weighted means are computed by aggregating over all segments:

$$\mu_x = \frac{\Sigma_{L\in\Omega} p_x(L)}{\Sigma_{L\in\Omega} len(L)}, \ \mu_y = \frac{\Sigma_{L\in\Omega} p_y(L)}{\Sigma_{L\in\Omega} len(L)} \quad (8)$$

Here, Ω represents the set of all straight-line segments connecting successive points within the same stroke.

To estimate dispersion, we compute the length-weighted squared deviation along the x-axis as

$$d_x(L) = \int_L (x - \mu_x)^2 dL = \frac{1}{3} len(L)[(x_1 - \mu_x)^2 + \quad (9)$$

$$(x_2 - \mu_x)^2 + (x_1 - \mu_x)(x_2 - \mu_x)] \quad (10)$$

The standard deviation along the x-axis can be estimated using the following formula:

$$\delta_x = \sqrt{\frac{\Sigma_{L\in\Omega} d_x(L)}{\Sigma_{L\in\Omega} len(L)}} \quad (11)$$

With all the information of μ_x, μ_y and δ_x estimated from one character, we can now normalize the coordinates by:

$$x_{new} = \frac{(x-\mu_x)}{\delta_x}, \ y_{new} = \frac{(y-\mu_y)}{\delta_y} \quad (12)$$

Notably, we use the same scale factor δ_x for both axes (rather than estimating δ_y) to preserve the original height-to-width ratio and maintain stroke directionality. After normalization, each character is mapped onto a standardized Cartesian coordinate system while preserving its overall shape and structural proportions.

3.6 Rendering coordinate sequence into images

Algorithm I: Rendering Algorithm for Converting Stroke Sequences to Grayscale Image. The procedure normalizes and scales the input stroke coordinates and renders the trajectory on a fixed-size grayscale canvas (see Appendix B, Fig. 8 for an example).

Step	Operation
Input	Input $S = \{s_1, s_2, \ldots, s_n\}$, image size, upscale factor u
Output	Grayscale image $I \in [0,1]^{size \times size}$
1	$h \leftarrow size \times u \mid m \leftarrow margin \times u, t \leftarrow line_{width} \times u$
2	$P \leftarrow \{(x,y) \mid x \geq 0, y \geq 0, (x,y,p) \in s, s \in S\}$
3	If $P \leftarrow= \emptyset$, *return white image* $I \in \mathbb{R}^{size \times size}, I(i,j) = 1$
4	$(min_x, max_x) \leftarrow (min_{(x,y)\in p} x, max_{(x,y)\in p} x)$ $(min_y, max_y) \leftarrow (min_{(x,y)\in p} y, max_{(x,y)\in p} y)$
5	$\Delta_x \leftarrow max_x - min_x + \epsilon, \Delta_y \leftarrow max_y - min_y + \epsilon$
6	$s \leftarrow h - 2m$
7	Initialize white canvas $H \in \mathbb{R}^{h\times h}$, with pixel values 255
8	**FOR EACH** substroke $s \in S$:
9	Set $prev \leftarrow None$
10	**FOR EACH** $point(x,y,p) \in s$:
11	**IF** $x < 0 \vee y < 0$: *set* $prev \leftarrow None$; *continue*
12	Normalize: $px \leftarrow m + \frac{(x - min_x)}{\Delta_x} . S$
13	$py \leftarrow m + \frac{(x - min_y)}{\Delta_y} . S$
14	**IF** $p = 0 \wedge prev \neq None$: draw line from $prev$ to (px, py)
15	Set $prev \leftarrow None$
16	**IF** $p = 1$: *set* $prev \leftarrow None$
17	**END FOR**
18	Resize $H \leftarrow I \in \mathbb{R}^{size \times size}$ using Lanczos resampling
19	Feature scale $I \leftarrow I/255.0$
20	**RETURN** I
21	**END FOR**

To prepare pen-based stroke sequences for CNN-based classification, each handwriting sample is transformed into a grayscale image (Algorithm I). Consider a stroke sequence $S = \{s_1, s_2, \dots, s_n\}$ where each sub stroke $s_i = \{(x_i, y_i, p_i)\}_{j=1}^{m_i}$ consists of a series of coordinate points paired with pen-state values. The pen-state $p_i \in \{0, 1, 2\}$ indicates whether the pen is in contact with the surface (pen-down) or lifted (pen-up).

The rendering process outputs an image with a target resolution $\in \mathbb{Z}_+$, typically set to 128×128 pixels. To enhance stroke clarity and apply anti-aliasing, the rendering is initially performed at a higher resolution determined by an integer upscale factor $\text{upscale\_factor} \in \mathbb{Z}_+$, then resized to the target resolution. Additional parameters include $\text{line\_width} \in \mathbb{Z}_+$, which defines the stroke thickness, and margin $\in \mathbb{Z}_+$, which allocates padding around the rendered content to prevent edge clipping. This rendering pipeline is implemented using the Python Imaging Library (PIL), which supports precise drawing operations and high-quality resampling methods such as Lanczos interpolation for final image resizing.

4. MODEL ARCHITECTURE

The KVerifyID framework is a hybrid Siamese network that verifies writers from two complementary online handwriting modalities: a grayscale rendered word image and its pen-trajectory sequence (x, y, p) (Fig. 5). The Siamese network consists of two identical, weight-sharing towers, one per sample in the pair. Each tower extracts modality-specific features, fuses them, and outputs a single 128-D writer embedding for similarity comparison.

The image branch encodes spatial patterns from the grayscale word rendering using stacked Conv2D–ReLU–MaxPool blocks:

- **Conv2D:** captures local spatial features. For example, a Conv2D layer $(1 \to 32, \text{kernel} = 5, \text{padding} = 2)$maps an input tensor $[B, 1, 64, 64]$ to $[B, 32, 64, 64]$.
- **MaxPool2D:** reduces spatial resolution while preserving salient structures. A 2×2MaxPool maps $[B, 32, 64, 64]$ to $[B, 32, 32, 32]$.
- **ReLU:** introduces nonlinearity after each convolution.

After three Conv2D–ReLU–MaxPool blocks, the feature map becomes $[B, 128, 8, 8]$. It is flattened to 8192 dimensions and projected via an MLP head Linear (8192→512→128) with ReLU and BatchNorm1d to produce a 128-D image embedding f_{img}. Dropout $(p = 0.3)$ is applied to reduce overfitting.

The coordinate branch encodes temporal writing dynamics from the sequence (x, y, p):

- A BiGRU with input size 3 and hidden size 64 per direction produces an output $[B, T, 128]$. The last time step (or an equivalent aggregation) yields a fixed-length vector $[B, 128]$ summarizing pen-trajectory dynamics.

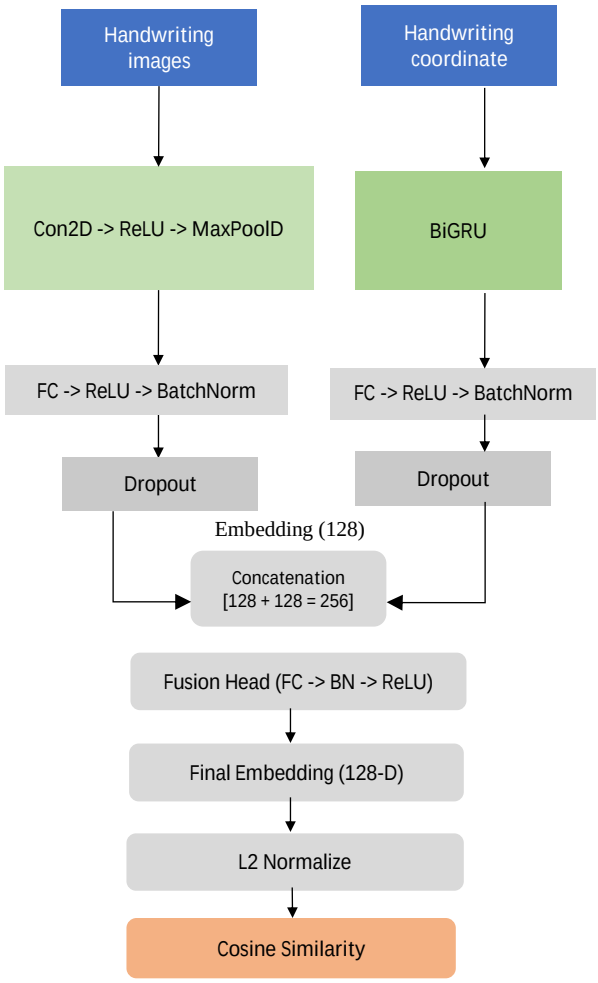

Fig. 5 The Overview of KVerifyID, a hybrid Siamese architecture with two weight-sharing towers. Each tower processes a handwritten word through two parallel streams: a CNN encoder for the grayscale rendered image (spatial cues) and a BiGRU encoder for the pen-trajectory sequence (x, y, p) (temporal dynamics). The resulting 128-D embeddings are concatenated $(256\text{-D})$and passed through a fusion head to produce a unified 128-D writer embedding. Writer similarity is computed using cosine similarity for the writer verification task.

- This vector is passed through a projection head Linear (128→256→128) with ReLU, BatchNorm1d, and Dropout to produce a 128-D coordinate embedding f_{coor}. Yielding $[B, 256]$. A lightweight fusion head (FC → BN → ReLU) then maps this vector to the final 128-D writer embedding. The embedding is L2-normalized and optionally scaled by a learnable factor (initialized to 10.0) to stabilize cosine-similarity training. Following prior Siamese verification work, this hybrid design integrates complementary spatial and temporal cues that are important for Khmer handwriting patterns [25].

The two modality embeddings are fused by concatenation:

$$f_{fusion} = [f_{img}; f_{coor}] \in \mathbb{R}^{256},$$

Writer similarity is computed using cosine similarity between two embeddings $f(x^i)$ and $f(x^j)$ is defined as:

$$S = cos\left(f(x^i), f(x^j)\right) = \frac{f(x^i).f(x^j)}{\|f(x^i)\|.\|f(x^j)\|} \quad (13)$$

Here,

$f(x^i) \in \mathbb{R}^n$ and $f(x^j) \in \mathbb{R}^n$ are the learned embedding vectors for the two inputs, $\|.\|$ denotes the ℓ_2-norm, and n is the embedding dimensionality. The similarity score satisfies $S \in [-1,1]$, where $S = 1$ indicates identical direction (maximum similarity), $S = 0$ indicates orthogonality, and $S = -1$indicates opposite direction (maximum dissimilarity).

For each pair of word instances (x_i, x_j), the two branches of the Siamese network produce 128-D embeddings $f(x_i)$ and $f(x_j)$. Their cosine similarity S is used as the verification score. Within the biometric evaluation protocol of Section III, a single global decision threshold τ is selected on the validation set (either at the equal-error-rate operating point, τ_{EER} or at the $FAR = 1\%$ operating point $\tau_{FAR} = 1\%$). At test time, this threshold τ is held fixed: a pair is classified as "same writer" if $S \geq \tau$ and classified as "different writer" otherwise. The resulting similarity scores are then used to compute the ROC curves, AUC, EER, FAR, FRR, accuracy, and F1-score.

5. RESULTS

5.1 Training environment setup

All procedures for data collection, preprocessing, experimentation, and evaluation were performed using the Python programming language. The deep learning models were developed and trained with the PyTorch framework. To maintain consistency and reproducibility across experiments, the following hyperparameters were employed:

Table 1. Training schedule and configuration of the KVerifyID Siamese model (total epochs and best validation epoch).

Model	Epochs	Best epoch (validation)	Backbone/ RNN Stack
GRU + LeNet	50	50	LeNet + GRU 1×64 (fusion)
LSTM + LeNet	50	49	LeNet + LSTM 1×64 (fusion)
LSTM + AlexNet	50	46	AlexNet + LSTM 1×64 (fusion)
BiLSTM + Simple CNN	50	48	Simple CNN + BiLSTM 1×64**)** (fusion)
KVerifyID	50	48	CNN + BiGRU (1 layer, hidden 64 per direction → 128-D)

- Machine: NVIDIA GeForce RTX 4090 GPU
- Learning rate: 0.0001
- Optimizer: AdamW ($lr\ 1 \times 10^{-4}$, weight decay 1×10^{-5}).
- Batch size: 32.
- Random seeds: {11, 33, 55}. We train each model with three random seeds and report one representative run, selected by the lowest validation EER. Unless otherwise stated, all metrics in Table 2 are from this selected run.
- Protocol: writer-disjoint splits; thresholds selected on validation and frozen for test.

Table 2. **Overall performance comparison on the writer-disjoint split. Results are reported for the best run among seeds {11, 33, 55}, selected by the lowest validation EER.**

Model	Train Acc (%)	Test Acc (%)	Val AUC (%)	Val EER (%)	FAR@ τ_{EER} (Val) (%)	FAR@EER (%)	FRR@ τ_{EER} (Val) (%)	FRR@ ERR (%)	Acc@1% FAR (%)	F1-Score (%)
GRU + LeNet	97.34	98.14	99.53	1.81	1.90	1.90	1.79	1.79	97.72	97.08
LSTM + LeNet	97.50	98.20	99.52	1.77	1.97	1.97	1.53	1.53	97.79	97.18
LSTM + AlexNet	99.39	99.26	99.86	0.69	0.78	0.78	0.68	0.68	99.28	99.09
BiLSTM + Simple CNN	99.58	99.46	99.92	0.59	0.59	0.59	0.47	0.47	99.31	99.13
KVerifyID	99.50	99.74	99.92	0.32	0.30	0.30	0.19	0.19	99.27	99.09

5.2 Evaluation metrics and protocol

We evaluated all systems under a writer verification (1:1) protocol. Each input is a pair of handwriting samples (x_i, x_j) labeled as genuine (same writer) or impostor (different writers). The Siamese network outputs a cosine-similarity

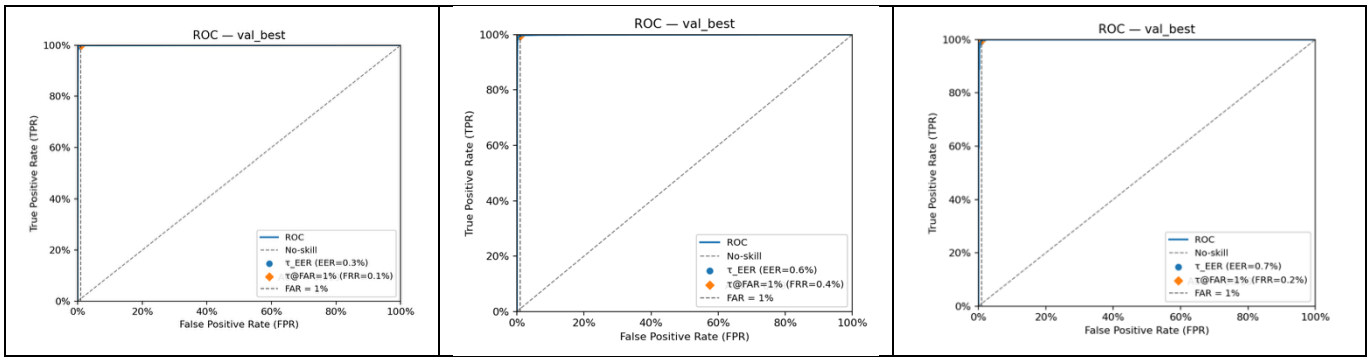

Fig. 6 Validation ROC curve of KVerifyID at the validation-best checkpoint. Markers indicate the validation-selected operating point at τ_{EER}. Reported KVerifyID performance: Train Acc = 99.50%, Test Acc = 99.74%, Val EER = 0.32%.

score $s \in [-1,1]$; a pair is accepted as same-writer if $s \geq \tau$ and rejected otherwise. All splits are writer-disjoint (train/validation/test) to avoid identity leakage. Classification outcomes as a function of the decision threshold τ:

- $TP(\tau)$: genuine (same-writer) pairs correctly accepted
- $FN(\tau)$: genuine pairs incorrectly rejected
- $TN(\tau)$: impostor (different-writer) pairs correctly rejected
- $FP(\tau)$: impostor pairs incorrectly accepted

Siamese embeddings:

$$z_a = f_\theta(x_a), z_b = f_\theta(x_b) \quad (14)$$

Cosine similarity

$$s = cos(z_a, z_b) = \frac{z_a^T z_b}{|z_a||_2 |z_b||_2} \quad (15)$$

Decision rule at threshold τ:

$$\hat{y}(\tau) = \begin{cases} 1, & s \geq \tau \\ 0, & s < \tau \end{cases} \quad (16)$$

Total numbers of genuine and impostor pairs

$$N_{pos} = TP(\tau) + FN(\tau), N_{neg} = TN(\tau) + FP(\tau) \quad (17)$$

Error rates

$$FAR(\tau) = \frac{FP(\tau)}{N_{neg}}, FRR(\tau) = \frac{FN(\tau)}{N_{pos}} \quad (18)$$

On the validation set, we sweep over thresholds $\boldsymbol{\tau}$ to obtain equal error rate $\boldsymbol{\tau_{EER}}$ threshold and a strict **FAR = 1%** operating point.

$$\tau_{EER} = \arg\min_{\tau} |FARval(\tau) - FRRval(\tau)| \quad (19)$$

On the validation set, we also choose a threshold that enforces approximately 1% FAR:

$$\tau_{FAR} = 1\% = \arg\min_{\tau} |FARval(\tau) - 0.01| \approx 0.99\ (\{s_i | (s_i, y_i) \in s_{val}, y_i = 0\}) \quad (20)$$

here $s_{val} = \{(s_i, s_j)\}$ is the validation set; s_i=Similar score; $y_i \in \{0,1\}$ (1=genuine, 0 = imposter); $Q0.99$ = empirical 99th percentile (right-continuous interpolation).

- Precision, Recall, Accuracy, F1 (used at τ_{EER} and τ_{FAR}): (21)

$$Acc(\tau) = \frac{TP(\tau)+TN(\tau)}{N_{pos}+N_{neg}} \quad (21a)$$

$$Precision\ (\tau) = \frac{TP(\tau)}{(TP(\tau)+FP(\tau))} \quad (21b)$$

$$Recall\ (\tau) = \frac{TP(\tau)}{TP(\tau)+FN(\tau)} \quad (21c)$$

$$F1\ (\tau) = \frac{2 \times Precision\ (\tau) \times Recall\ (\tau)}{Precision\ (\tau) + Recall\ (\tau)} \quad (21d)$$

Figures mark the validation-selected operating points: the equal-error threshold τ_{EER} and the $\tau_{FAR=1\%}$ line. Tables report metrics at these fixed thresholds. Unless otherwise noted, results are reported for the single selected run (chosen by the lowest validation EER among seeds {11, 33, 55}). The best epoch is selected by the lowest validation EER.

5.3 Result and comparison

All experiments follow the writer-verification (1:1) protocol with writer-disjoint train/validation/test splits. We train only KVerifyID and select the best checkpoint by the lowest validation EER. A decision threshold $\tau_{EER}$is selected once on validation and then frozen for test evaluation. As summarized in Table 2, KVerifyID achieves Train Acc = 99.50% and Test Acc = 99.74%. Verification error on validation is EER = 0.32%. At the validation-selected equal-error threshold τ_{EER}, the corresponding operating errors are FAR = 0.30% and FRR = 0.19%, indicating a strong balance between false accepts and false rejects under the fixed-threshold protocol.

6. DISCUSSION AND CONCLUSION

6.1 Discussion

This work evaluates online Khmer writer verification (1:1) using KVerifyID, a hybrid dual-stream Siamese model, under writer-disjoint train/validation/test splits. The model

is selected by the lowest validation EER, and the decision threshold τ_{EER} is chosen on validation and then frozen for test evaluation. Under this protocol, KVerifyID achieves 99.74% test accuracy with low verification error (0.32% validation EER) and low operating errors at τ_{EER}(FAR = 0.30%, FRR = 0.19%). These results indicate that fusing spatial word structure with temporal stroke dynamics provides a reliable basis for discriminating writers in Khmer script under text-independent, word-level conditions.

We also observe that BiLSTM + Simple CNN attains slightly higher training accuracy and Acc@1% FAR, which may reflect stronger fitting to the training distribution. In contrast, KVerifyID delivers the best overall generalization on the writer-disjoint evaluation (highest test accuracy and lowest validation EER), supporting the benefit of multimodal fusion for robustness when verifying unseen writers.

6.2 Conclusion

We formulated online Khmer handwriting as a writer-verification (1:1) problem and proposed KVerifyID, a hybrid dual-stream Siamese architecture that fuses rendered word images with pen-trajectory sequences (x, y, p). Under strict writer-disjoint splits with validation-fixed thresholding, KVerifyID achieves 99.50% training accuracy and 99.74% test accuracy, with Val EER = 0.32% and operating errors FAR = 0.30% and FRR = 0.19% at τ_{EER}. These results demonstrate robust generalization to unseen writers and support KVerifyID as an effective approach for Khmer online writer verification in authentication and forensic screening scenarios.

7. FUTURE WORK

Future work will extend KVerifyID beyond pairwise verification toward deployment-oriented scenarios. We will evaluate few-shot enrollment (e.g., $k \in \{1,3,5\}$) by forming writer templates from averaged embeddings and measuring FRR at fixed FAR. We also plan to study cross-device and cross-session generalization by collecting additional sessions and heterogeneous hardware conditions, and to expand the dataset with more writers, words, and writing conditions. On the modeling side, we will explore stronger metric-learning objectives (e.g., ArcFace/CosFace or supervised contrastive learning), improved fusion strategies, and efficiency techniques such as pruning, quantization, and distillation for practical deployment.

ACKNOWLEDGEMENTS

The authors would like to express their sincere gratitude to the ViLa Lab for its generous support, insightful feedback, and access to essential resources, which significantly contributed to improving the quality of this research. The authors also acknowledge the Institute of Technology of Cambodia (ITC) for providing scholarship support. In addition, the authors gratefully acknowledge Heng Samrin Thbong Khmum University (UHST) for its affiliation and institutional support, including the opportunity to pursue Ph.D. study at ITC.

REFERENCES

[1] M. Javidi and M. Jampour, “A deep learning framework for text-independent writer identification,” Eng. Appl. Artif. Intell., vol. 95, Art. no. 103912, Oct. 2020.

[2] Z. Li, M. Jiang, Q. Wang, and H. Li, “Deep template matching for offline handwritten Chinese character recognition,” J. Eng., vol. 2020, no. 4, pp. 120–124, Apr. 2020, doi: 10.1049/joe.2019.0895.

[3] N. Kimlong and V. Dona, “Online handwriting identification from Khmer script using a deep learning approach with KhNet,” in Proc. 14th Conf. Inf. Technol. Its Appl. (CITA), Jan. 2026, pp. 751–763, doi: 10.1007/978-3-032-00972-2_55.

[4] M. Javidi and M. Jampour, “Handwriting analysis: Psychological aspects and pattern recognition,” Pattern Anal. Appl., vol. 23, no. 4, pp. 899–912, Nov. 2020.

[5] C. C. Tappert, C. Y. Suen, and T. Wakahara, “The state of the art in online handwriting recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 12, no. 8, pp. 787–808, Aug. 1990.

[6] L. Likforman-Sulem, A. Esposito, M. Faundez-Zanuy, S. Clemençon, and G. Cordasco, “EMOTHAW: A novel database for emotional state recognition from handwriting,” arXiv preprint arXiv:2202.12245, 2022. [Online]. Available: https://arxiv.org/abs/2202.12245

[7] V. K. Seema and R. Shilpa, “Recognition of emotional state based on handwriting analysis and psychological assessment,” Int. J. Eng. Adv. Technol. (IJEAT), vol. 8, no. 6, pp. 4395–4402, Aug. 2019.

[8] C. E. Chaski, “Who’s at the keyboard? Authorship attribution in digital evidence investigations,” Int. J. Digit. Evid., vol. 4, no. 1, pp. 1–13, 2005.

[9] M. A. M. Hasan, J. Shin, and M. Maniruzzaman, “Online Kanji characters based writer identification using support vector machine,” Appl. Sci., vol. 12, no. 20, Art. no. 10249, Oct. 2022.

[10] C. Adak, B. B. Chaudhuri, and M. Blumenstein, “An empirical study on writer identification and verification from intra-variable individual handwriting,” IEEE Access, vol. 7, pp. 24738–24758, Feb. 2019, doi: 10.1109/ACCESS.2019.2899908.

[11] Z. Y. He and Y. Y. Tang, “Chinese handwriting identification using convolutional neural networks,”

IEEE Trans. Cybern., vol. 54, no. 2, pp. 678–690, Feb. 2024.
[12] B. Purkaystha, T. Datta, and M. S. Islam, "Bengali handwritten character recognition using deep convolutional neural network," in Proc. 20th Int. Conf. Comput. Inf. Technol. (ICCIT), Dhaka, Bangladesh, 2017, pp. 1–5.
[13] B. Annanurov and N. M. Noor, "Handwritten Khmer text recognition," in Proc. IEEE Int. WIE Conf. Electr. Comput. Eng. (WIECON-ECE), Pune, India, 2016, pp. 176–179, doi: 10.1109/WIECON-ECE.2016.8009112.
[14] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: A comprehensive survey," IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 1, pp. 63–84, Jan. 2000, doi: 10.1109/34.824821.
[15] M. Bulacu and L. Schomaker, "Text-independent writer identification and verification using textural and allographic features," IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 4, pp. 701–717, Apr. 2007.
[16] S. Dey, A. Dutta, J. I. Toledo, S. K. Ghosh, J. Lladós, and U. Pal, "SigNet: Convolutional Siamese network for writer-independent offline signature verification," arXiv preprint arXiv:1707.02131, 2017.
[17] N. Kimlong, V. Dona, and P. Vannaro, "A deep learning approach for identifying individuals based on their handwriting," Techno-SR J., vol. 13, no. 1, pp. 52–58, Jul. 2024.
[18] L. Mengsay, "Handwritten Khmer digit recognition," Towards Data Science (Medium), 2019. [Online]. Available: https://medium.com/towards-data-science/handwritten-khmer-digit-recognition-860edf06cd57. [Accessed: Jun. 2020].
[19] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a 'Siamese' time delay neural network," in Advances in Neural Information Processing Systems (NeurIPS), vol. 6, 1994, pp. 737–744.
[20] W. Yang, L. Jin, and M. Liu, "DeepWriterID: An end-to-end online text-independent writer identification system," IEEE Intell. Syst., vol. 31, no. 2, pp. 45–53, Mar./Apr. 2016, doi: 10.1109/MIS.2016.22.
[21] A. Graves, Supervised Sequence Labelling with Recurrent Neural Networks. Berlin, Germany: Springer, 2012.
[22] S. Dey, A. K. Bhunia, S. Pal, U. Pal, and P. P. Roy, "Signet-Bengali and Signet-Indic: Two benchmark datasets for offline signature verification in Indic scripts," IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 9, pp. 4717–4731, Sep. 2022, doi: 10.1109/TPAMI.2021.3053561.
[23] H. Sasagawa, "The establishment of the national language in twentieth-century Cambodia: Debates on orthography and coinage," *Southeast Asian Stud.*, vol. 4, no. 1, pp. 43–72, Apr. 2015.
[24] J.-M. Filippi and C. Vatho, *The Linguistic Atlas of Cambodia and Standard Khmer Language*. Phnom Penh, Cambodia: Royal Univ. Phnom Penh, 2016. [Online]. Available: https://www.researchgate.net/publication/359083929_The_Linguistic_Atlas_of_Cambodia_and_Standard_Khmer_Language. [Accessed: Mar. 2016].
[25] M. W. A. Kesiman *et al*., "Benchmarking of document image analysis tasks for palm leaf manuscripts from Southeast Asia," *J. Imaging*, vol. 4, no. 2, Art. no. 43, 2018.
[26] Python Software Foundation, "Python language reference," Python Documentation. [Online]. Available: https://docs.python.org/3/reference/index.html. [Accessed: Aug. 3, 2024].
[27] P. Thakur and P. Jadon, "Django: Developing web using Python," in *Proc. ICACITE*, 2023, pp. 303–306.
[28] Django Software Foundation, "Django web framework documentation (FAQ)," Django Documentation. [Online]. Available: https://docs.djangoproject.com/en/5.0/faq/general/. [Accessed: Aug. 3, 2024].

APPENDIX A
OTKH Tool Development and Data Collection

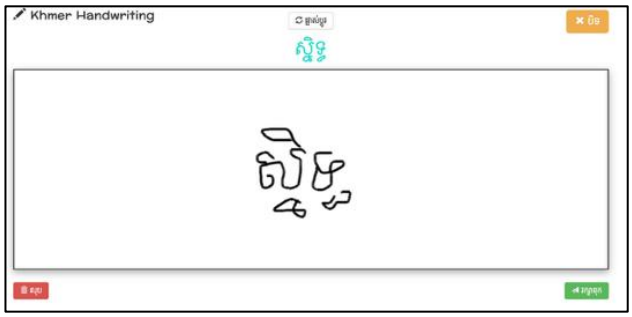

Fig. 7 User interface for drawing Khmer scripts using OTKH. The black word (ស្ទឹង) represents the user's handwritten input, while the blue word above represents the word generated from the database (photograph by author). The word drawn by the user is represented as a series of points (x, y).

To support online Khmer handwriting acquisition, we developed a web-based platform called Online Tools for Khmer Handwriting (OTKH). The tool enables users to register, log in, and write Khmer words directly in a browser-based drawing interface. During writing, the system records each sample as a time-ordered trajectory of (x, y)coordinates (with pen-state when available), together with the corresponding word label and writer metadata.

Although handwriting produced on a digital surface can differ from paper due to reduced tactile feedback and device-specific input characteristics, core writer-specific motor patterns such as stroke order, pen-lifts, and movement rhythm are typically preserved across writing environments. Nevertheless, data quality may be influenced by factors such as input device type, screen size, and user familiarity with the interface. In early usage, users may show higher variability due to a short learning period; as familiarity increases, writing behavior generally stabilizes, improving consistency. Repeated interactions can further reduce variation, although fatigue and other external factors may still introduce subtle changes in stroke patterns.

To develop the OTKH platform, Python a widely adopted high-level, general-purpose programming language introduced by programming language (van Rossum, 1991) in an opensource a Python-based [26] was used in conjunction with Django [27], an open-source Python-based web framework designed to streamline web application development and minimize redundant coding tasks. Django provides a modular structure that supports rapid development and deployment by leveraging pre-built components. As noted by Adrian Holovaty et al. [28], Django follows the Model-View-Template (MVT) architectural pattern, which is conceptually aligned with the Model-View-Controller (MVC) design framework. The MVT architecture separates application logic into three main components:

- Model: defines the data schema and manages storage/retrieval in the database.
- View: handles request processing and application logic.
- Template: renders the user interface and dynamically displays content to users.

Using OTKH, we collected 4,878 instances of handwritten words from 298 participants, with each participant contributing an average of approximately 16 word samples. The resulting database stores the online coordinate sequences, word labels, and associated writer attributes, providing the foundation for the writer verification experiments in this study.

APPENDIX B
Coordinate Normalization Example

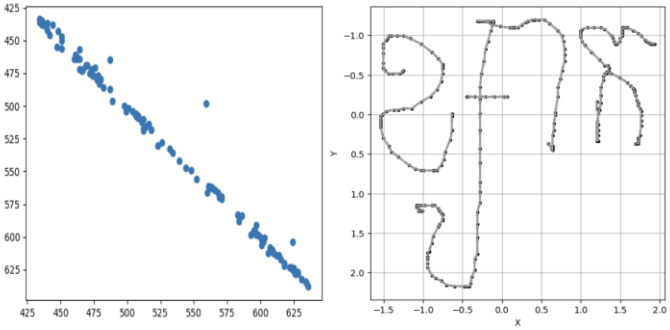

Fig. 8 Coordinate normalization example: (a) raw online stroke coordinates before normalization; (b) the same sample after centering and scaling using (μ_x, μ_y) and (δ_x, δ_y), illustrating preserved structure with standardized scale and alignment.

This appendix provides an illustrative example of the coordinate normalization procedure used in our preprocessing pipeline. Fig. 8 compares the same online Khmer word sample before and after normalization. The raw trajectory is first centered using the estimated means (μ_x, μ_y) and then scaled using the corresponding standard deviations (δ_x, δ_y). As shown, normalization standardizes scale and alignment across samples while preserving the overall word structure, which is important for consistent rendering and model training.

APPENDIX C

DET Curves (same model sets). Validation Det for The Same Configurations; Dashed Vertical Line Indicates far = 1%.

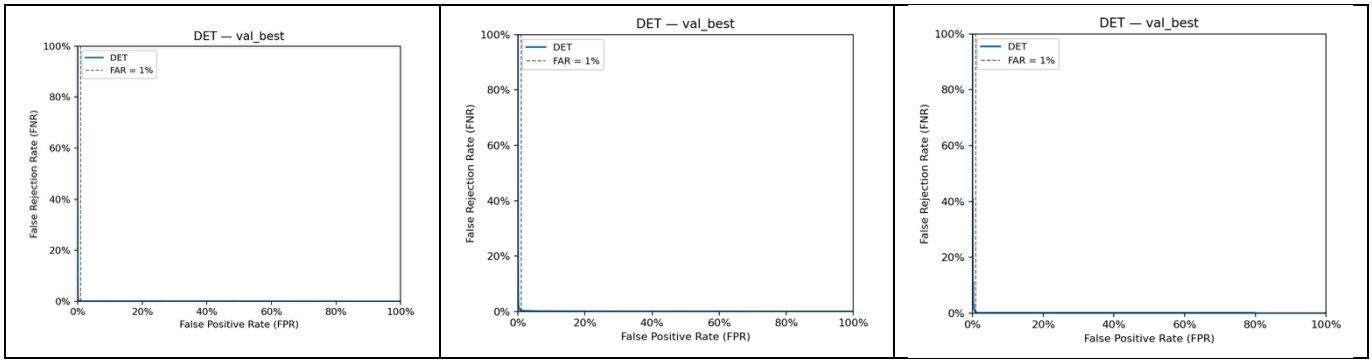

Fig. 9 DET curves (FNR vs. FPR, log-scaled axes style) for VGG-16, BiLSTM, LSTM+SimpleCNN, and KhmerWriterID at *val_best*. The vertical dashed line marks FAR = 1%. These plots complement the ROC view by expanding the low-error region around the 1% FAR operating point used in the paper.

APPENDIX D

Score histograms. Positive (same-writer) vs. Negative (different-writer) Similarity Distributions at The Validation-best Checkpoint for Representative Models; Titles Annotate μ_{pos} And μ_{neg}.

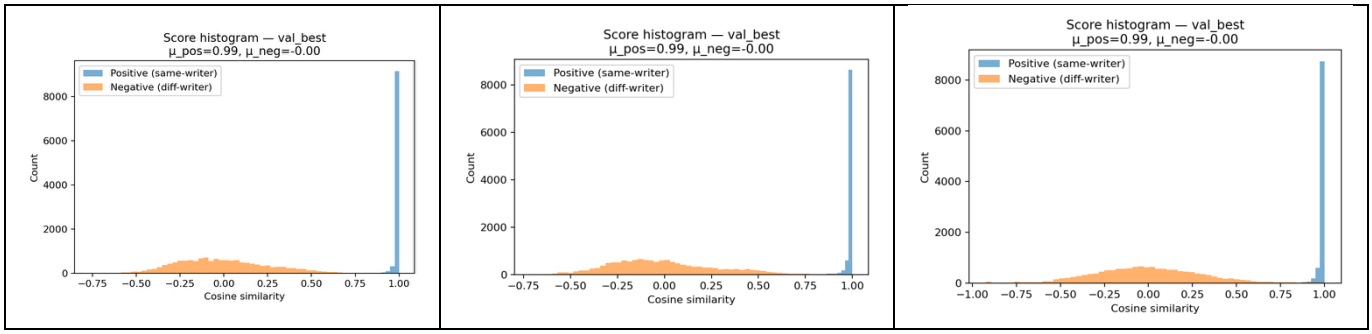

Fig. 10 Cosine-similarity score histograms for genuine (same-writer) and impostor (different-writer) pairs at val_best for VGG-16, BiLSTM, LSTM+SimpleCNN, and KhmerWriterID. Mean scores μ_{pos} and μ_{neg} are shown above each panel. Larger separation and a tighter positive peak indicate stronger verification margins at the chosen operating thresholds.