Lightweight Siamese Neural Network for Offline Signature Verification: A Data-Efficient Approach

Zoltán Dominik Boros* and Gábor Kertész*†‡
*Obuda University John von Neumann Faculty of Informatics, Budapest, Hungary
†Laboratory of Parallel and Distributed Systems, Institute for Computer Science and Control (SZTAKI)
Hungarian Research Network (HUN-REN), Budapest, Hungary
‡Corresponding author, e-mail: kertesz.gabor@nik.uni-obuda.hu

2025 IEEE 29th International Conference on Intelligent Engineering Systems (INES) | 979-8-3315-9771-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/INES67149.2025.11078027

***Abstract*—Signature verification is essential in banking, judicial, and governmental institutions. This study presents an offline signature authentication system based on a lightweight Siamese Neural Network (SNN) with metric learning. Unlike large CNN-based based approaches that rely on large datasets and complex architectures, this method utilizes extensive preprocessing and a compact CNN architecture to reduce computational cost while maintaining competitive accuracy. The images are pre-processed before being fed to the two-headed architecture; the system based on denoised, cropped and centered images achieved accuracy of 80-85% on the CEDAR and BHSig260 datasets, while using approximately 1.93 million parameters, significantly fewer than deep CNN-based approaches like state-of-the-art SigNet (10M+ parameters). The results demonstrate that effective preprocessing and hyperparameter tuning can enable data-efficient signature verification without requiring large-scale datasets.**

***Index Terms*—signature verification, siamese neural network, metric learning, contrastive learning, image preprocessing**

I. Introduction

Signature forgery is one of the most common and recognized methods of identity fraud, highlighting the critical need for robust signature authentication systems to distinguish between genuine and forged signatures.

The authentication of signatures holds significant importance in banking, legal frameworks, and security systems, as well as in the daily lives of individuals. Furthermore, even legitimate users may occasionally fail to reproduce their signature consistently. Historically, signature verification was a manual process handled by professionals such as bank clerks or postal workers, prone to human error [1]. With technological advancements, automating this process has become increasingly viable.

Signature verification systems perform two primary functions: verifying the legitimacy of a signature, which is the authentication step. The difficult part is to match a signature to an individual, which step is referred to as identification. These systems leverage databases to compare input signatures with stored samples, determining their validity or assigning them to a specific individual. Online systems utilize dynamic properties [2] such as speed and pressure captured during signature creation; offline systems, which analyze static handwritten signatures, are particularly useful in document-based workflows where real-time data capture is not possible.

Offline signature verification remains essential in many traditional workflows, this project aims to develop a machine learning-based solution for authentication and identification.

Metric learning is a machine learning paradigm that trains models to learn a similarity function by mapping input samples to an embedding space, where similar instances are placed closer together while dissimilar instances are pushed apart. Contrastive learning is a specific metric learning method that optimizes a model by comparing paired samples; the model minimizes the distance between similar pairs and maximizes the distance between dissimilar pairs.

The aim of this study is to apply metric learning to discriminate genuine and forged signatures and ensure reliable identification and reduce risks of fraud. Unlike most deep learning-based signature verification systems that rely on large datasets and deep networks, this approach utilizes a lightweight CNN with extensive preprocessing and metric learning, making it computationally efficient and data-efficient.

II. Related Work

The field of signature verification has been extensively studied, ranging from handcrafted feature extraction to deep learning-based methods. Offline signature verification, which analyzes static images of handwritten signatures, has received significant attention due to its relevance in paper-based workflows.

Signature verification has evolved from early handcrafted feature extraction methods to modern deep learning-based approaches. While traditional techniques rely on geometric and statistical features [3], they struggle with high intra-class variability, due to their inability to generalize across different handwriting styles [4]. Deep learning methods address this limitation by automatically learning robust features from raw signature images [5], reducing the need for manual feature engineering.

A. Similar Projects

One of the foundational studies in contrastive learning-based signature verification was conducted by Bromley et al. [6], introducing the Siamese Time Delay Neural Network. This system utilized a digital signature capture device to record dynamic features such as pen speed and pressure, coupled with preprocessing steps like size normalization and noise filtering.

By testing two architectures with varying layer configurations, the study introduced Siamese Neural Networks (SNNs), a shared-weight architecture that can effectively learn similarity metrics for handwritten signatures.

Subsequent research emphasized writer-independent systems to improve scalability by consolidating data into a unified model [7]. These approaches use SNNs with twin Convolutional Neural Networks (CNNs) to learn feature representations. A one-shot learning approach further optimized these models by minimizing data requirements and enabling threshold-based classification.

A more recent approach by Xiao and Ding employed focal-loss functions to address data imbalance in signature verification systems [8]. This model combined original and enhanced input images, utilizing convolutional layers and modular data weighting to improve feature extraction and classification across diverse datasets.

Other advancements in the field include triplet loss-based solutions, where genuine, forged, and reference signatures were used to optimize intra-class and inter-class distances [9]. Unlike contrastive loss, which considers only paired inputs, triplet loss optimizes feature embeddings using three samples: an anchor, a positive, and a negative [10]. This approach refines intra-class compactness and inter-class separation. Similarly, CNN-based systems incorporated preprocessing techniques such as noise reduction and normalization to enhance input quality, extended with classification using Support Vector Machines (SVMs) [11]. A deep CNN model featuring 18 layers extended this approach, focusing on edge detection and so-called critical feature extraction [12].

Wei et al. [13] introduced the Inverse Discriminative Network along with a novel collection of datasets for chinese signatures. The proposed architecture is based on the siamese architecture, and extends it by adding inverse images and comparing them pair-by-pair. The resulting network provides three decisions, that are merged into a single output. This approach singificantly reduced the False Rejection Rate (FRR), and slightly improved accuracy on generally accepted benchmark datasets, such as CEDAR [14]. The authors also published the collected datasets under the name BHSig.

SigNet by Dey et al. [15] is a well-known baseline in signature verification research. Utilizing SNNs with Deep CNNs as twin sub-networks, it applies contrastive loss and Euclidean distance for classification. Preprocessing includes resizing, inversion, and normalization, while architectural features like Local Response Normalization to enhance performance and generalization across datasets. It is worth mentioning that while SigNet is based on a deep CNN trained on large datasets, our proposed approach applies more extensive preprocessing and a smaller neural network, maintaining competitive accuracy with fewer parameters.

III. Methodology

This section describes the methodology used to implement the offline signature verification system based on image preprocessing, siamese-architectured CNN using contrastive loss based on euclidean distance in the embedded space.

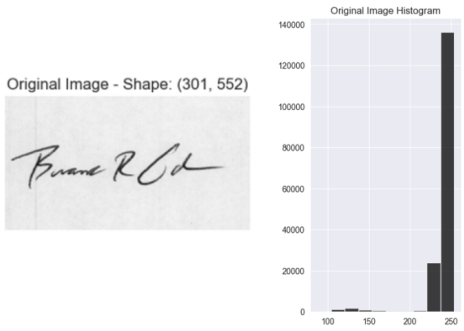

Fig. 1. The original image and corresponding histogram after loading.

A. Image preprocessing

Since signature images exhibit variations due to noise, distortion, and inconsistent alignment, preprocessing is essential to standardize inputs and enhance model performance. The preprocessing pipeline is implemented using OpenCV, which provides efficient image manipulation functions necessary for noise reduction and feature enhancement. The preprocessing pipeline consists of six distinct steps:

- **Image Reading:** The signature image is loaded in 8-bit grayscale format for subsequent processing (Fig. 1).
- **Inversion:** Each pixel value is inverted (for 8-bit `uint8` representation, technically the value is subtracted from 255). This ensures that the signature is represented in white on a black background, a format suitable for further operations.
- **Noise Reduction:** Gaussian blur is applied with a 5×5 kernel to smooth the image. This is followed by Otsu's thresholding for binarization. This step reduces noise and ensures a clean input for the model (Fig. 2).
- **Cropping:** The signature region is isolated on the denoised image using contour detection, a bounding box with a 5-pixel margin is applied to remove irrelevant background; the result is a tightly cropped image.
- **Center Alignment:** The cropped image is resized and placed on a blank canvas of predefined dimensions while preserving its aspect ratio. This step avoids distortion during resizing and ensures the signature is centered on the canvas (Fig. 3).
- **Normalization and Tensor Conversion:** Before feeding the image into the model, all pixel values are normalized to the range [0, 1] (for `uint8` representation it is done by dividing by 255). The result is converted into a tensor format.

This preprocessing pipeline standardizes input images, reducing noise and structural variations, enhancing feature extraction and model robustness in signature verification.

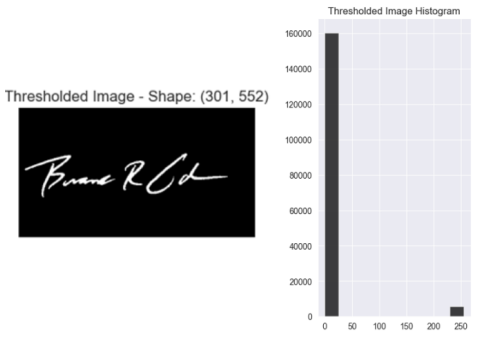

Fig. 2. The resulting image and corresponding histogram of the noise reduction step.

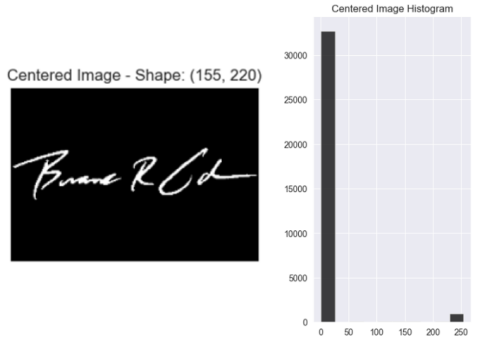

Fig. 3. Cropped and centered image and corresponding histogram.

B. Model Architecture

The custom CNN backbone architecture consists of five convolutional blocks followed by dense layers, designed to extract hierarchical feature representations for signature verification. These feature embeddings are later used in a Siamese framework for similarity comparison. The architectural details are detailed in Table I and visualized on Fig 4.

The convolutional backbone contains five blocks of convolutional and pooling layers, extracting spatial and structural features from input signatures. A global average pooling (GAP) layer flattens these representations, which are then processed by two fully connected layers to generate compact feature embeddings.

1) Siamese Architectured Neural Network: The Siamese Neural Network (SNN) is based on two identical CNNs edscribed previously. Each CNN processes one of the input signature images, producing a representation in embedded space. These extracted feature vectors are compared using a distance layer to quantify similarity.

Euclidean distance [16] is chosen as the distance function as it captures absolute spatial differences in feature embeddings, which is crucial in this specific task. Unlike cosine similarity, which measures angular similarity and ignores magnitude, Euclidean distance ensures that small variations in signature structure and spatial alignment are reflected in the embedding space. Additionally, Euclidean distance aligns naturally with contrastive loss, it is a typical choice for metric learning

TABLE I
DETAILED ARCHITECTURE OF THE CNN BACKBONE USED IN THE SNN. NOMENCLATURE FOLLOWS TENSORFLOW / KERAS TERMINOLOGY.

Layer/Block	***Details***	***Key Parameters***	***Output Shape***
Block 1	Conv2D	Filters: 32, Kernel: $(7, 7)$, Stride: $(1, 1)$	$149 \times 214 \times 32$
	LeakyReLU	Alpha: 0.3	–
	MaxPooling2D	Pool size: $(2, 2)$, Stride: $(2, 2)$	$74 \times 107 \times 32$
Block 2	Conv2D	Filters: 64, Kernel: $(5, 5)$, Stride: $(1, 1)$	$70 \times 103 \times 64$
	BatchNormalization	–	–
	LeakyReLU	Alpha: 0.3	–
	MaxPooling2D	Pool size: $(2, 2)$, Stride: $(2, 2)$	$35 \times 51 \times 64$
Block 3	Conv2D	Filters: 128, Kernel: $(3, 3)$, Stride: $(1, 1)$	$33 \times 49 \times 128$
	BatchNormalization	–	–
	LeakyReLU	Alpha: 0.3	–
	MaxPooling2D	Pool size: $(2, 2)$, Stride: $(2, 2)$	$16 \times 24 \times 128$
Block 4	Conv2D	Filters: 256, Kernel: $(3, 3)$, Stride: $(1, 1)$	$14 \times 22 \times 256$
	LeakyReLU	Alpha: 0.3	–
	Dropout	Rate: 0.2	–
Block 5	Conv2D	Filters: 512, Kernel: $(3, 3)$, Stride: $(1, 1)$	$12 \times 20 \times 512$
	LeakyReLU	Alpha: 0.3	–
	MaxPooling2D	Pool size: $(2, 2)$, Stride: $(2, 2)$	$6 \times 10 \times 512$
	Dropout	Rate: 0.2	–
Global Pooling	GlobalAveragePooling2D	–	1×512
Dense Layer 1	Dense	Units: 512, Kernel Reg.: $1e^{-4}$	–
	Dropout	Rate: 0.5	–
Dense Layer 2	Dense	Units: 128, Kernel Reg.: $1e^{-4}$	–

based solutions. The fixed-scale interpretation allows simple thresholding for verification tasks, making it an optimal choice for signature-based authentication.

Euclidean distance can be formulated as:

$$\mathcal{D}(s_1, s_2) = \sqrt{\sum_{i=1}^{n} (f(s_1)_i - f(s_2)_i)^2}, \tag{1}$$

where $f(s_1)$ and $f(s_2)$ are the feature vectors of the input images s_1 and s_2 from the CNN, and n is the dimensionality of the feature space.

The contrastive loss function is applied to optimize the SNN. It ensures that similar pairs remain close in the embedding space (intra-class variance is reduced); while dissimilar pairs are separated by at least a margin m (inter-class separation is increased). Contrastive loss can be defined as:

$$\begin{aligned} \mathcal{L}(s_1, s_2, y) =& (1 - y)\frac{1}{2}\mathcal{D}(s_1, s_2)^2 + \\ & y\frac{1}{2}\{\max(0, m - \mathcal{D}(s_1, s_2))\}^2, \end{aligned} \tag{2}$$

where:

- s_1, s_2: Input signature images.
- y: Binary label ($y = 0$ for genuine-genuine pairs and $y = 1$ for genuine-forged pairs).
- m: Margin value defining the minimum distance for dissimilar pairs.
- $\mathcal{D}$: Euclidean distance defined in equation (1).

In this implementation, pairs from the same source ($y = 0$) are optimized to have a smaller Euclidean distance, ensuring their embeddings are closer in the feature space. On the other hand, negative pairs (where $y = 1$) are penalized if their distance is less than the margin m, pushing their embeddings farther apart.

2) Optimization and Training: The RMSprop optimizer was chosen due to its ability to adaptively adjust learning rates per parameter, helping stabilize training in the contrastive learning setting. Adam can perform similarly, however it is

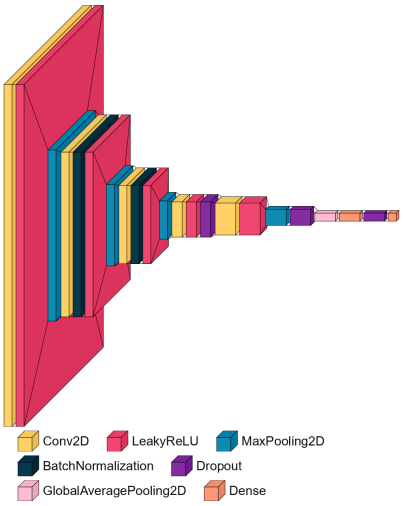

Fig. 4. Diagram of the CNN backbone used in the SNN, illustrating the convolutional layers and feature extraction pipeline.

more powerful when performing updates which can cause instability in the embedding space. RMSprop ensures smoother convergence of feature representations.

Standard stochastic gradient descent approach often converges more slowly for contrastive loss, RMSprop is efficient in separating embeddings in the metric space [17].

To improve generalization, early stopping was applied to abort training when validation performance stopped improving; also, a learning rate scheduler reduced the learning rate by a factor of 0.1 upon stagnation.

IV. Experimental Setup and Training

This section describes the experimental setup, including dataset preparation, training procedures. The performance of the model is measured using multiple metrics. Additionally, t-Distributed Stochastic Neighbor Embedding (t-SNE) [18], [19] was employed to visualize feature embeddings, and confusion matrix offers detailed metrics.

A. Dataset

Three publicly available datasets were used for evaluation: CEDAR, BHSig260-Bengali, and BHSig260-Hindi. To ensure robust training, datasets were balanced by including an equal number of genuine-genuine and genuine-forged pairs. The CEDAR dataset, comprising 55 authors, yielded 30.360 pairs (552 per author) after balancing. The BHSig260-Bengali dataset, with 100 authors, resulted in 55.200 balanced pairs, while the BHSig260-Hindi dataset, with 160 authors, contained 88.320 pairs. For all datasets, 80% of the pairs were used for training, and the remaining 20% were reserved to test performance. From the training subset 10% of the samples were split as validation set, elements not directly used in training, instead used to detect overfitting. This approach ensured that training, validation, and testing sets were completely independent, with no overlap of authors across splits.

TABLE II
Balanced Pair Distribution for Training, Validation, and Testing

Dataset	Authors	Training Pairs	Validation Pairs	Test Pairs
CEDAR	55	21,859	2,429	6,072
BHSig-Bengali	100	39,744	4,416	11,040
BHSig-Hindi	160	63,590	7,066	17,664

B. Implementation

Tensorflow and OpenCV was applied during implementation using Python.

Signature images were resized to $155 \times 220 \times 1$, which is the input dimensionality of the backbone CNN. Each batch contained 32 signature pairs, with an equal distribution of genuine-genuine and genuine-forged samples, ensuring balanced learning during training.

To improve generalization and robustness to signature variability, data augmentation was applied during training. Transformations included random horizontal flips to account for minor writing distortions, and brightness and contrast adjustments to simulate variations in document scanning conditions. These transformations introduced variability in the training data while preserving essential features of the signatures.

The model was optimized using the contrastive loss function, with a margin set to $m = 1.2$. The threshold distance was set to 0.5 during evaluation as decision boundary. The RMSprop optimizer was employed with an initial learning rate of $1e^{-4}$.

Regularization techniques, such as dropout and kernel regularization, were implemented to prevent overfitting. Dropout rate was set to 0.5 in fully connected layers, and to 0.2 in convolutional layers. Weight-decay based L2 kernel regularizer was applied with a factor of $\lambda = 1e^{-4}$.

Early stopping was configured to monitor the validation loss, terminating training if no improvement for three consecutive training steps. Additionally, the `ReduceLROnPlateau` callback was used to adjust the learning rate dynamically, reducing it by a factor of 0.1 if the validation loss did not improve after two epochs, down to a minimum of $1e^{-6}$.

Training was conducted for a maximum of 20 training steps. It is worth mentioning that instead the word ”epoch”, ”training step” is used, as during training a subset of all possible combinations were used. The final model was selected based on the lowest validation loss.

V. Results and Evaluation

This section presents the evaluation results of the proposed SNN across three datasets: CEDAR, BHSig260-Bengali, and BHSig260-Hindi. Performance was assessed using training, validation, and testing accuracy, along with t-SNE visualizations to analyze feature embedding separability. The impact of dataset characteristics, particularly noise in CEDAR and dataset scale in BHSig260, is discussed.

The CEDAR dataset posed the greatest challenge due to significant noise and scanning artifacts, resulting in testing

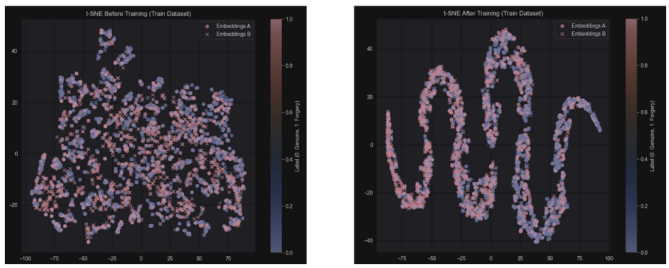

Fig. 5. t-SNE visualization before-after training on the training set (CEDAR)

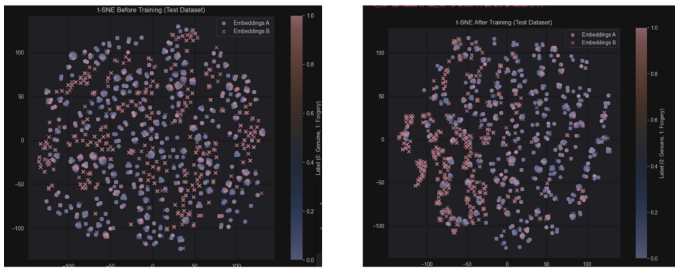

Fig. 6. t-SNE visualization before-after training on the testing set (CEDAR)

accuracy of 80%. Validation accuracy reached 80%, while training accuracy peaked at 99%. The t-SNE visualizations (Fig. 5 and Fig. 6) illustrate the clustering of genuine and forged pairs before and after training, indicating the model's capacity to distinguish between the classes effectively.

The BHSig260-Bengali dataset yielded a validation accuracy of 85% and a testing accuracy of 84%, highlighting the model's robustness when applied to clean and balanced data. Similarly, the BHSig260-Hindi dataset achieved validation accuracy of 88% and testing accuracy of 87%, underscoring the model's ability to handle larger datasets effectively. In both datasets, training accuracy remained at 99%, suggesting that the model successfully learned the underlying patterns without overfitting, as evidenced by consistent validation and testing results.

A. Further results

Although one-shot learning was not the primary focus of this study, additional experiments were conducted to assess the

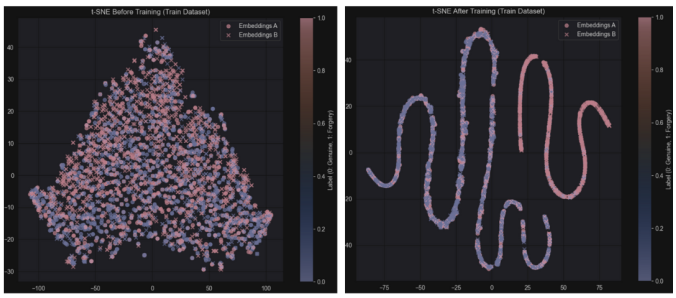

Fig. 7. t-SNE visualization before-after training on the training set (BHSig260-Hindi)

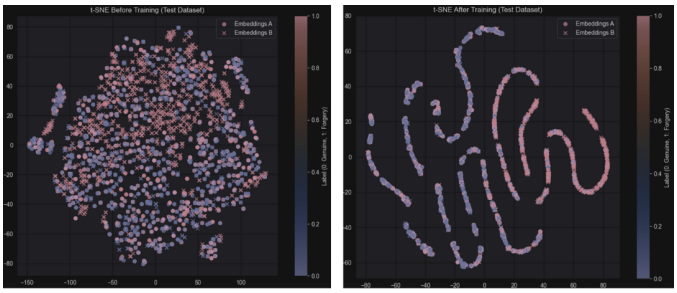

Fig. 8. t-SNE visualization before-after training on the testing set (BHSig260-Hindi)

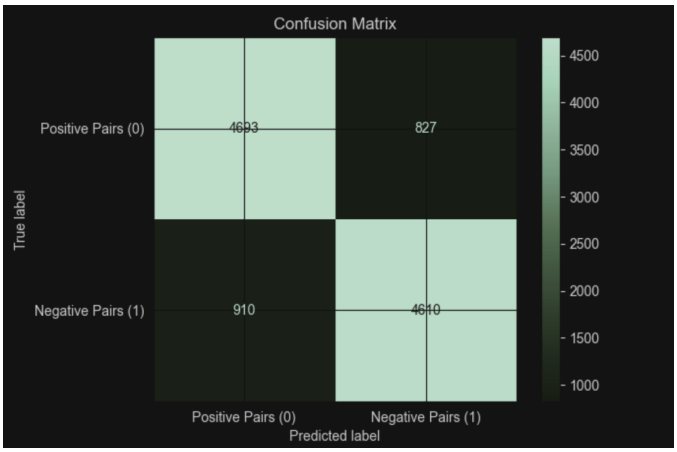

Fig. 9. The confusion matrix on the BHSig260-Hindi dataset

model's ability to recognize unseen signatures using distance-based one-shot testing. In one-shot testing, a single reference signature from the test dataset is compared against $N-1$ signatures belonging to different authors. This method primarily demonstrates the model's capability for recognition tasks, such as character recognition, even though it is not the primary focus of this implementation.

In this approach, a parameter k determines the number of test runs. The test is considered successful if the positive pair produces the smallest distance among the comparisons. The function calculates the success rate as a percentage based on the ratio of successful runs to total runs.

It is worth noting that the inherent prediction process of the model on unseen data can also be regarded as a form of one-shot testing, as the network evaluates completely novel signature pairs during inference.

TABLE III
SUMMARY OF MODEL PERFORMANCE ACROSS DATASETS, COMPARING TRAINING, VALIDATION, AND TESTING ACCURACY.

Dataset	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
CEDAR	99	80	80
BHSig260-Bengali	99	85	85
BHSig260-Hindi	99	88	87

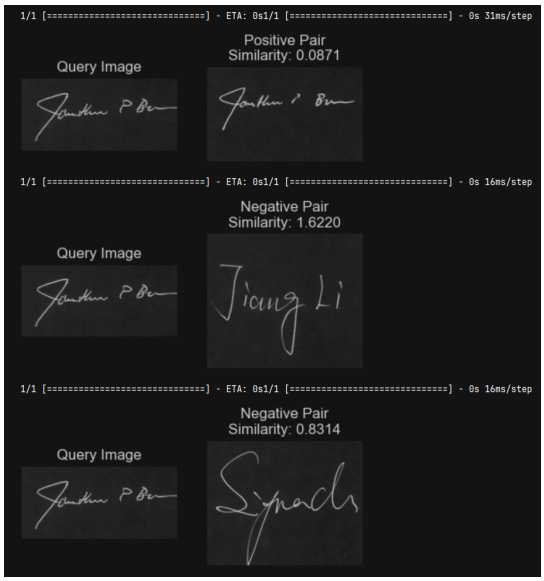

Fig. 10. Example output of the one-shot testing

VI. Conclusion

The implemented model demonstrated consistently strong performance across all tested datasets, achieving up to 87% accuracy on BHSig260-Hindi and maintaining robust performance even on challenging datasets like CEDAR (80%). The method achieves comparable accuracy to larger architectures like SigNet while using significantly fewer training samples and a more computationally efficient model, making it suitable for resource-constrained environments.

The primary objective of distinguishing between genuine and forged signatures relative to a reference was achieved with high accuracy. Although the model was not primarily designed for one-shot learning, additional experiments confirmed its ability to generalize to unseen signatures.

One area requiring further refinement is the overfitting observed during training. Despite applying regularization techniques (dropout, L2 weight decay) and hyperparameter tuning, further optimization is needed. Future work could explore increasing data augmentation diversity, adjusting dropout rates, or experimenting with contrastive loss variations to improve generalization [20].

Acknowledgement

The authors express their gratitude to the members of the Applied Machine Learning Research Group at Obuda University John von Neumann Faculty of Informatics for their valuable comments and suggestions.

References

[1] J.-J. Brault and R. Plamondon, "A complexity measure of handwritten curves: Modeling of dynamic signature forgery," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 2, pp. 400–413, 1993.

[2] D. Guru and H. Prakash, "Online signature verification and recognition: An approach based on symbolic representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 6, pp. 1059–1073, 2008.

[3] R. Plamondon and G. Lorette, "Automatic signature verification and writer identification—the state of the art," *Pattern recognition*, vol. 22, no. 2, pp. 107–131, 1989.

[4] E. J. Justino, A. El Yacoubi, F. Bortolozzi, and R. Sabourin, "An off-line signature verification system using hmm and graphometric features," in *Proc. of the 4th international workshop on document analysis systems*, pp. 211–222, Citeseer, 2000.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[6] 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*, vol. 6, 1993.

[7] M. V. Arısoy, "Signature verification using siamese neural network one-shot learning," *International Journal of Engineering and Innovative Research*, vol. 3, no. 3, pp. 248–260, 2021.

[8] W. Xiao and Y. Ding, "Two-stage siamese network model for offline handwritten signature verification," *Symmetry*, vol. 14, no. 12, p. 1216, 2022. doi:10.3390/sym14121216.

[9] H. Rantzsch, H. Yang, and C. Meinel, "Signature embedding: Writer independent offline signature verification with deep metric learning," in *Advances in Visual Computing: 12th International Symposium, ISVC 2016, Las Vegas, NV, USA, December 12-14, 2016, Proceedings, Part II 12*, pp. 616–625, Springer, 2016.

[10] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[11] E. Alajrami, B. A. M. Ashqar, B. S. Abu-Nasser, A. J. Khalil, M. M. Musleh, A. M. Barhoom, and S. S. Abu-Naser, "Handwritten signature verification using deep learning," *International Journal of Academic Multidisciplinary Research (IJAMR)*, vol. 3, no. 12, pp. 39–44, 2019. http://www.ijeais.org/ijamr.

[12] A. B. Jagtap, R. S. Hegadi, and K. C. Santosh, "Feature learning for offline handwritten signature verification using convolutional neural network," *International Journal of Technology and Human Interaction (IJTHI)*, vol. 15, pp. 54–62, October 2019. https://ideas.repec.org/a/igg/jthi00/v15y2019i4p54-62.html.

[13] P. Wei, H. Li, and P. Hu, "Inverse discriminative networks for handwritten signature verification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5764–5772, 2019.

[14] M. K. Kalera, S. Srihari, and A. Xu, "Offline signature verification and identification using distance statistics," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 07, pp. 1339–1360, 2004.

[15] S. Dey, A. Dutta, J. I. Toledo, S. K. Ghosh, J. Llados, and U. Pal, "Signet: Convolutional siamese network for writer independent offline signature verification." https://arxiv.org/abs/1707.02131, 2017.

[16] H. Baltzakis and N. Papamarkos, "A new signature verification technique based on a two-stage neural network classifier," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 1, pp. 95–103, 2001.

[17] F. Wang and H. Liu, "Understanding the behaviour of contrastive loss," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2495–2504, 2021.

[18] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

[19] M. F. Mayda and A. Musdholifah, "Siamese-network based signature verification using self supervised learning," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 17, no. 2, pp. 115–126, 2017.

[20] D. T. Várkonyi, D. T. Bányai, and A. R. Várkonyi-Kóczy, "Investigating traditional machine learning models and the utility of audio features for lightweight swarming prediction in beehives," *Acta Polytechnica Hungarica*, vol. 21, no. 10, 2024.