GMR The Original

Lightweight And Low-Latency Inference Based Online Signature Verification

***Sarvabhatla Mrudula[1] , Dr Ravi Kumar Tata[2]**

[1]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India 2302031034@kluniversity.in
[2]Associate Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India rktata5860@gmail.com

ABSTRACT
An online signature can be viewed as a set of multivariate time-series signals, and it remains a widely used biometric for real-time user verification on memory-constrained devices such as digital signature pads. Recent advances in deep learning (DL) have encouraged its adoption for Online Signature Verification (OSV); however, most DL-based OSV solutions still inherit major deployment challenges: (i) heavyweight models with lakhs of trainable parameters, (ii) limited feasibility for resource-constrained edge devices, and (iii) high inference latency in real-time usage. To address these limitations while retaining the benefits of DL, this work leverages TensorRT, a parallel programming model built on NVIDIA CUDA, to perform post-training optimization of DL models and to accelerate inference. We experimentally analyze the above OSV bottlenecks and then present a superior framework that achieves state-of-the-art (SOTA) performance on two widely used datasets, MCYT-100 and SVC. In particular, the proposed model reports an equal error rate (EER) of 11.98% and 5.03% in the Skilled_01 category for the MCYT-100 and SVC datasets, respectively. The results further confirm that the proposed approach produces lightweight models with reduced inference latency: a 74.76% reduction in model size leads to only a 2.22% reduction in accuracy in the Skilled_01 category of MCYT. Overall, the experimental findings show that the proposed framework yields optimized OSV models that are lightweight, low-latency, and accurate, making them suitable for deployment on resource-constrained edge devices.

Keywords : Online Signature Verification, Deep Learning, Model Compression, Lightweight Model, low-latency Inference.

1. INTRODUCTION

1. Introduction

Advances in wireless and networking technologies have resulted in the widespread adoption of online signatures as a commonly used biometric for authentication in mobile commerce, administrative, and financial applications [39], [41], [14], [44]. Based on the acquisition process, signature verification systems are categorised into offline and online methods. Offline systems rely on scanned images of handwritten signatures, whereas online signature verification (OSV) captures dynamic, pressure-sensitive temporal features such as velocity, pressure, and acceleration using digital devices like tablets and stylus pens [33], [34].

With the rapid growth of deep learning (DL) as a dominant paradigm for solving complex AI problems [31], [3], [6], [15], DL techniques have recently been adopted for OSV, which was traditionally addressed using classical machine learning approaches [32], [1], [38], [39], [41], [26], [27], [14]. Despite their improved accuracy, DL-based OSV models suffer from key limitations, including large model sizes, high inference latency, and poor suitability for resource-constrained edge devices. Moreover, existing studies primarily focus on training-time optimization, while post-training efficiency remains underexplored [8], [43].

Recent works have explored advanced deep learning paradigms such as mobile-based biometric systems [44], multimodal authentication combining signature and facial features [46], and transformer-based architectures for signature modeling [52], [53]. While these approaches improve performance, they often introduce additional computational complexity, making them less suitable for real-time deployment on edge devices.

To address these challenges, this work employs TensorRT [20], a high-performance inference optimisation framework that applies post-training optimisations such as precision calibration, layer and tensor fusion, and kernel auto-tuning, enabling lightweight and low-latency OSV models suitable for real-time edge deployment.

Recent progress in deep learning (DL) has improved OSV performance over traditional methods; however, DL-based OSV models are often computationally intensive, incur high inference latency, and are unsuitable for deployment on resource-constrained edge devices. Moreover, existing works largely focus on training-time optimization, leaving post-training efficiency underexplored. To address these limitations, this work adopts TensorRT, a CUDA-based inference optimization framework, to perform post-training model optimization through precision calibration, layer fusion, and kernel auto-tuning, enabling lightweight, low-latency OSV systems suitable for real-time edge deployment.

Motivation and major contribution of this paper:

In this work, we propose an OSV framework in which a trained DL baseline model is converted, post-training, into a TensorRT (TRT) model. After conversion, TensorRT applies a sequence of inference-time optimizations, which are detailed in the subsequent sections, to reduce model footprint and accelerate execution while preserving verification performance. Overall, the key contributions of this paper can be summarized as follows: Exploring the idea of pruning the model post training using TensorRT deep learning platform.

- Efficient calibration selection, i.e., FP32, FP16, INT8, based on the comparative analysis of EER outcome by the baseline model and corresponding calibrated models.
- Comparative analysis of inference time by the baseline and various calibrated models.
- Thorough experimental analysis and evaluation of the proposed framework with two publicly used datasets 1. MCYT-100 2. SVC.

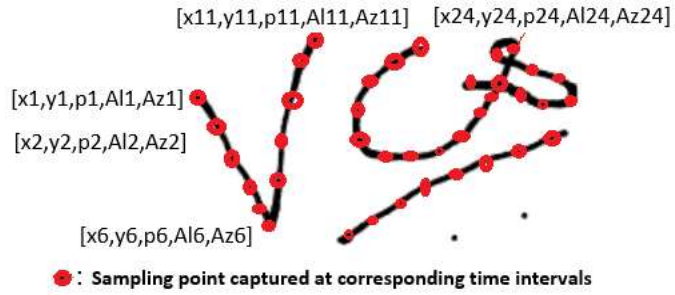

Fig. 1. An online signature is represented as a collection of sample points. Each sample point is represented as a set of five profiles: [x-coordinate, y-coordinate, pressure (p), Altitude (Al), Azimuthal angle (Az)].

The rest of the paper is structured as follows: Section 2 presents the Multi-Teacher-Student OSV framework; Section 3 details the proposed Curriculum Learning algorithm; Section 4 presents experimental analysis and comparisons; and Section 5 concludes the work.

Note : Manuscript typed with computer using Times New Roman font (Doc) one Column, on paper size 21 cm x 29.7 cm (A4), 1 spaced and in 11 point Times New Roman font. There should be a top and a bottom margin of 3.0 cm, with right and left margins of length 3 cm.

2. Literature Review

Until recently, OSV research was initially dominated by classical approaches such as dynamic time warping (DTW) [28], [30], [22], symbolic representations [10], and Gaussian mixture models [29]. Other widely explored directions include symbolic representations [10], [17], Gaussian Mixture Models (GMMs) [29], template matching methods [21], [22], and feature-engineering strategies such as anthropomorphic feature-based techniques [18] and stability region-based modeling [7]. More recently, Subash et al. introduced a series of studies based on machine learning classifiers and ensemble techniques [34], [33]. A comprehensive survey covering OSV progress over the past decade is available in [19]. Since this work focuses on DL-based OSV, we next review the most relevant developments in that direction.

An early DL-based study [23] explored sequence modeling using an LSTM to capture the temporal dynamics of online signature signals. Although the authors investigated multiple settings—including user-specific modeling, recognition, and verification—the reported performance was not competitive. Deep learning approaches later emerged, including LSTM-based sequence models [23], Siamese architectures [40], and autoencoder-based models [1]. However, these methods showed limited consistency across forgery types. Additional sequence-model-driven OSV methods include [32] and [1]: the former leverages gated autoregressive units, whereas the

latter employs an LSTM autoencoder. Notably, [32] reported stronger results on random forgeries, while [1] demonstrated better robustness in skilled-forgery scenarios.

Although sequence models are a natural choice for capturing the temporal nature of online signatures, they have not consistently delivered markedly superior OSV performance. More recently, starting around 2019, convolutional neural networks (CNNs) have been increasingly explored for OSV and have generally proven more effective than purely sequence-based approaches [41], [26], [25], [27], [14], [16]. For example, [41] integrates a Siamese CNN with DTW: the Siamese network learns discriminative representations from signature sequences, while a DTW module performs temporal alignment between the two signatures. In [25], separable convolutions were shown to be computationally efficient and effective, even under challenging few-shot learning settings. Building on this, [27] extends [25] by fusing deep features (extracted using a hybrid CNN–LSTM architecture) with handcrafted statistical features. A related fusion strategy is investigated in [26], but it extracts deep features using a convolutional autoencoder, and its training pipeline does not rely on an LSTM component. Additionally, [14] proposed a 1D-CNN operating on fixed-length representations of online signatures and explored the use of synthetic forgeries to reduce or eliminate dependence on skilled forgeries during training. Recent advancements include multimodal biometrics [46], federated learning-based OSV [49], and transformer-based architectures [52]. Advanced feature modeling using dynamic and kinematic properties has also been explored [48], [51], along with hybrid deep learning approaches [53].

Most of the DL-based OSV frameworks discussed above do not explicitly address model compression or deployment-oriented optimization. As a result, they inherit common limitations of deep models—namely large parameter counts, increased memory footprint, and higher inference latency, which restrict practical usage on resource-constrained platforms. To mitigate these issues, this work proposes a TensorRT-based OSV framework that targets post-training optimization to reduce model size and accelerate inference, while maintaining verification performance. The proposed framework is summarized in the following sections.

3. Proposed OSV Framework

In this section, we describe the key steps of the proposed approach in detail. As illustrated in Fig. 2(a), a conventional deep learning workflow typically comprises designing the model architecture, training the network, and saving the trained model (weights) after training. During inference, the saved model is loaded and used to generate predictions on test data or real-time inputs. In contrast, as shown in Fig. 2(b), converting a trained baseline model—irrespective of the original training framework—into a TensorRT model enables optimized inference execution. This conversion improves runtime efficiency by increasing throughput (i.e., the number of input samples processed per unit time) and reducing end-to-end latency. The overall pipeline consists of a sequence of steps, which we summarize below:

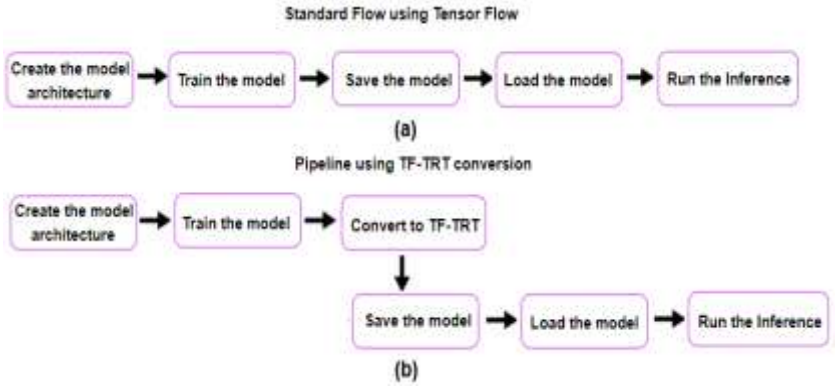

Fig. 2. a) the pipeline for standard train and inference flow for Tensorflow model. b) the pipeline for TF-TensorRT converted models.

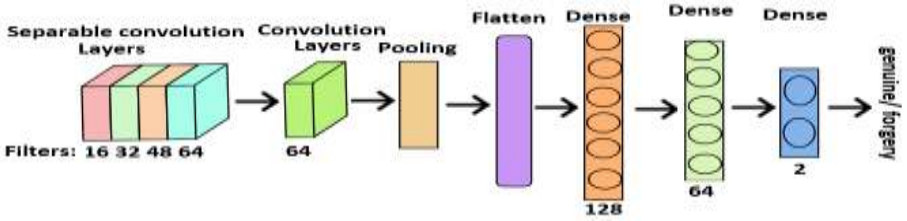

Fig. 3. The architecture for baseline model.

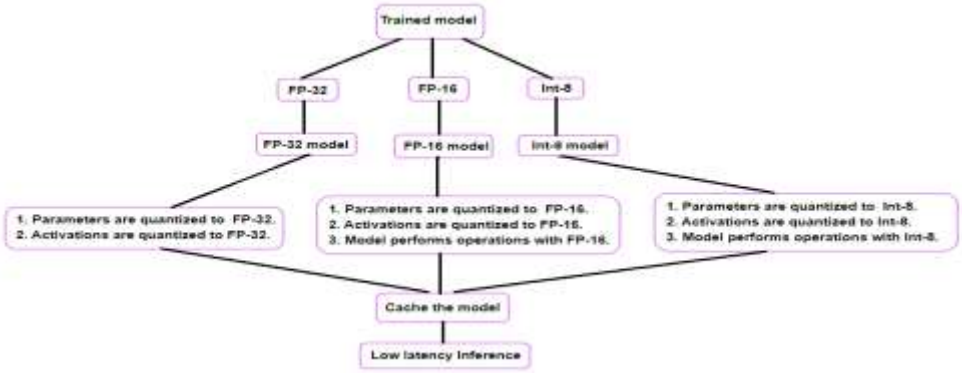

Fig. 4. The maximum, average and optimal length of signatures of each writer in case of MCYT-100 data-set.

- Create and train an FP32 Tensorflow or any other framework-based baseline model.
- Compute the baseline model throughput and accuracy
- Convert the baseline model into a TF-TRT Float32 model and compute the throughput and accuracy
- Convert the baseline model into a TF-TRT Float16 model and compute the throughput and accuracy
- Convert the baseline model into a TF-TRT INT8 model and compute the throughput and accuracy.

We describe each step in detail as follows:

Create and train a FP32 Tensorflow-based baseline model:

The proposed baseline architecture is illustrated in Fig. 3. It comprises four 1D depthwise separable convolution layers with 16, 32, 48, and 64 filters and kernel sizes of 1, 3, 5, and 7, respectively. For MCYT-100, each online signature is represented as a fixed-length feature vector of size 1×100, whereas for the SVC dataset the input dimensionality is 1×47; in both cases, this vector is provided as input to the first separable convolution layer. The output of the fourth separable convolution block is then passed to a standard 1D convolution layer with 64 filters of size 1×3. The resulting feature maps are processed through a max-pooling operation and subsequently flattened. The flattened representation (of size 1×271) is finally fed into a classifier consisting of two fully connected layers followed by a softmax output layer, with 128 and 64 neurons, respectively.

Compute the Baseline model Throughput and Accuracy

After defining the model architecture, the network is trained on the training set using the hyperparameters listed in Table 1. Upon completion of training, the resulting FP32 baseline model is evaluated on the test set. The corresponding test accuracy is recorded together with the model size, as reported in Table 2. Finally, the best-performing model weights are saved for subsequent use during inference.

Convert the baseline model into a TF-TRT Float32, Float 16 and Int 8 model and compute the throughput and accuracy

The baseline network is trained and stored in FP32 precision, meaning that its weights, biases, and intermediate activations are represented using 32-bit floating-point values. As shown in Fig. 4, we then convert this baseline model into TensorRT-accelerated variants—namely TF-TRT FP32, FP16, and INT8—using TensorRT [20]. This conversion produces optimized, memory-efficient runtime engines that improve inference throughput. During the FP16/INT8 conversion, TensorRT performs a calibration procedure to approximate the original FP32 model while minimizing information loss and quantization noise. After conversion, computations are performed at the target precision (FP32/FP16/INT8) rather than using full FP32 throughout. In addition, TensorRT applies graph-level optimizations such as layer fusion, combining compatible operations to reduce kernel launches and memory transfers, thereby improving latency without substantially impacting accuracy. Overall, precision calibration and layer fusion lead to: (i) reduced inference time (as reported in Tables 3 and 4), (ii) accuracy that remains comparable to, or close to, the FP32 baseline, and (iii) improved memory utilization that supports loading and running multiple networks efficiently during inference. Finally, the converted engines are cached for reuse in the inference stage, further reducing end-to-end runtime.

During conversion to TF-TRT FP32, FP16, and INT8 (lower-precision execution), the effective numerical representation of layer weights can be constrained by the reduced dynamic range—particularly in FP16, whose representable range is smaller than FP32—potentially leading to saturation/overflow effects and value truncation. Nevertheless, our experimental results indicate that these reduced-precision effects do not materially degrade

verification performance; as reported in Tables 2 and 6, the optimized TensorRT variants maintain accuracy close to the FP32 baseline.

Table 1 - The set of hyper parameters used in the proposed baseline model.

Parameter	Value
Optimizer	0.0002
Learning Rate(r)	4
Batch Size	1 e-08
beta_2	0.999
beta_1	0.9
Decay	0.01
epoch	100

4. Experimental Analysis

We comprehensively evaluate the proposed OSV framework on two widely used datasets, namely MCYT-100 [9] and SVC [35], with summary statistics reported in Table 6. All experiments were conducted on an NVIDIA Titan-X GPU (12 GB). Performance is assessed across five training regimes: Skilled_01 (S_1), Skilled_05 (S_5), Skilled_10 (S_10), Skilled_15 (S_15), and Skilled_20 (S_20). The training and testing protocol is defined as follows. Consider a dataset with W users, where each user has R genuine signatures and F forged signatures. For a given Skilled_C (S_C) setting, the framework is trained using C genuine and C forged signatures per user, and is evaluated on the remaining $R-C$ genuine and $F-C$ forged samples per user.

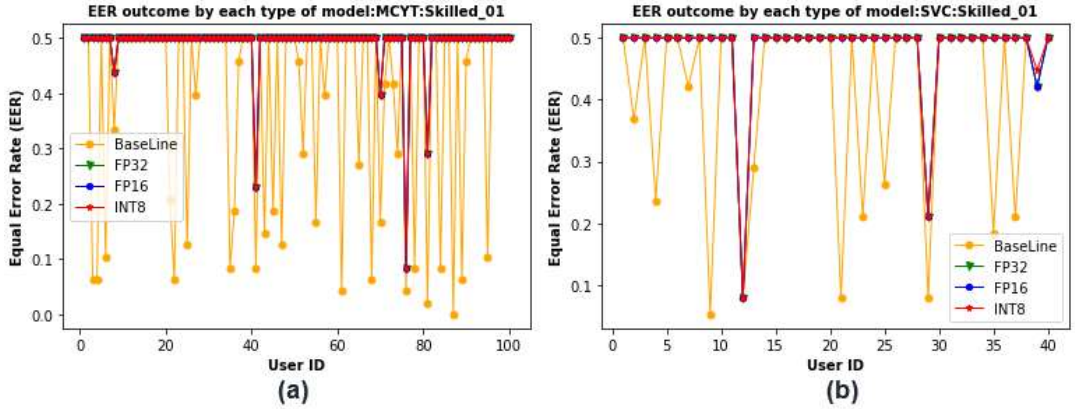

Fig. 5. The EER outcome by the model proposed in [25], in case of a) one shot learning Skilled_01: MCYT-100 and b) Skilled_01: SVC

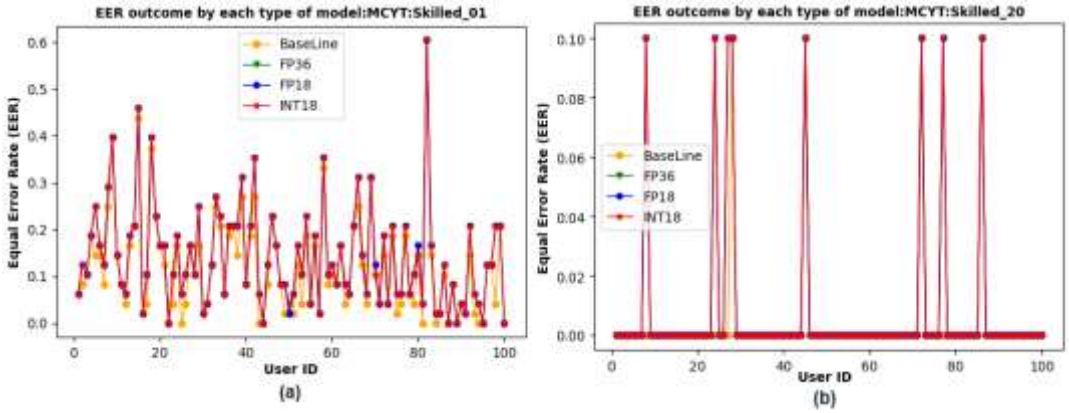

Fig. 6. The EER outcome by the baseline and corresponding calibrated model, in case of a) one shot learning (Skilled_01) and b) Skilled_20.

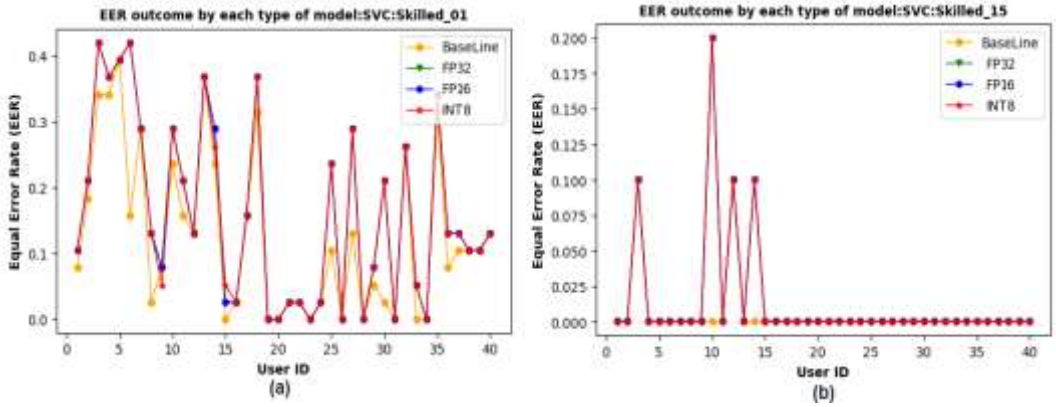

Fig. 7. The EER outcome by the baseline and corresponding calibrated model, in case of a) one-shot learning (Skilled_01) and b) Skilled_20.

Table 2 Comparison of the model size of Baseline and other representative models on MCYT dataset.

	Baseline	FP32	FP16	Int8
Proposed: Model Size	416 KB	348.48 KB	348.48 KB	104.98 KB
Model Size [25]	518 KB	467.69 KB	467.69 KB	124.53 KB

Table 3 Comparison of the model size of Baseline and other representative models on SVC dataset.

	Baseline	FP32	FP16	Int8
Proposed: Model Size	416 KB	348.48 KB	348.48 KB	104.98 KB
Model Size [25]	518 KB	467.69 KB	467.69 KB	124.53 KB

Table 4 Comparison of the inference time (in milli seconds) of the baseline and other representative models for various experimental categories: MCYT-100. The values enclosed in brackets indicates the inference time for the model proposed in [25]

	Baseline	FP32	FP16	Int8
Skilled_01	103.47 (103.40)	99.74 (99.66)	99.77 (99.70)	98.98 (97.90)
Skilled_05	105.60	99.80	99.81	99.00
Skilled_10	104.32	99.79	99.81	99.20
Skilled_15	108.10	99.85	99.85	99.46
Skilled_20	110.32	99.87	99.88	99.72

Table 5 Comparison of the inference time (in milli seconds) of the baseline and other representative models for various experimental categories: SVC. The values enclosed in brackets indicates the inference time for the model proposed in [25]

	Baseline	FP32	FP16	Int8
Skilled_01	41.23 (41.22)	39.94 (39.92)	39.94 (39.91)	39.79 (39.41)
Skilled_05	41.25	39.95	39.94	39.83
Skilled_10	41.25	39.95	39.95	39.87
Skilled_15	42.23	39.96	39.96	39.93

The EER results obtained from the various experimental settings are reported in Tables 7 and 8. For clarity, the best (lowest) EER in each column is denoted by "*", and the second-best is denoted by "**". We further note that our framework is evaluated consistently across a broad range of training categories, enabling a more comprehensive comparison than studies that report results for only a limited subset of settings. As shown in the subsequent analysis, the proposed approach maintains strong performance even in the challenging one-shot learning regime, i.e., the Skilled_01 category.

As reported in Table 2, the baseline model in the proposed framework has a size of 416 KB, while its INT8 optimized variant requires only 104.98 KB. In comparison, the baseline model in [25] is 518 KB, and its corresponding INT8 version is 124.53 KB. In terms of verification performance, Table 7 shows that our baseline achieves an EER of 11.98%, outperforming the baseline in [25], which reports 13.42%. A similar trend is observed on SVC: Table 3 indicates that our baseline model is 216 KB, whereas the model in [25] is 318.61 KB; after INT8 optimization, the model sizes reduce to 61.98 KB (proposed) and 65.32 KB ([25]), respectively. The corresponding performance comparison in Table 8 further confirms the advantage of our approach, with the proposed baseline achieving an EER of 5.03% versus 5.83% for [25].

Additionally, as shown in Table 2, the FP32 and FP16 TensorRT representations of the proposed model remain compact, with sizes of 348.48 KB and 178.28 KB for MCYT-100 and SVC, respectively. Runtime behavior is summarized in Tables 4 and 5, where the INT8 variants consistently achieve the lowest inference time across both datasets; the bracketed values correspond to the inference time reported for [25]. Although [25] occasionally exhibits slightly lower latency, its EER remains higher than that of the proposed framework (Table 7), and the same relationship is reflected in the SVC comparison (Table 5 and Table 8).

To analyze user-wise behavior under extreme data scarcity, Fig. 5(a) presents per-user EER for each model representation under the one-shot learning setting (Skilled_01), and also reports the Skilled_01 outcomes of [25] on both MCYT-100 and SVC. Notably, except for the baseline, the calibrated variants in [25] show an EER degradation of approximately 0.5%. In contrast, Figs. 6(a) and 7(a) demonstrate that our proposed framework maintains strong one-shot performance, with average EER of approximately 0.2%–0.3% on MCYT-100 and 0.3%–0.4% on SVC, which is substantially lower than [25]. As the number of training samples increases (e.g., 20 samples per user), Fig. 6(b) shows that all models achieve lower EER compared to Fig. 6(a); in particular, the INT8 model yields the lowest EER, producing zero EER for most users, with only a small subset (eight users) exhibiting higher error rates. Similar trends are also observed in Fig. 7(b).

5. Comparison with SOTA Methods

Tables 7 and 8 summarize the comparative performance of the proposed method against representative approaches from the literature on both datasets. A key observation from these tables is that many existing methods report results only for a subset of the evaluation categories, and such limited reporting can restrict conclusions about real-time applicability and robustness across operating conditions. In contrast, our framework is evaluated consistently across a broad range of skilled categories, enabling a more comprehensive assessment. The results in Tables 7 and 8 demonstrate that the proposed approach performs strongly across all skilled settings on both datasets. Notably, even in the most challenging one-shot learning regime (Skilled_01), the proposed framework achieves a performance gain of over 1.28% on MCYT-100 and over 0.34% on SVC-Task 2 relative to prior work.

Table 6 Details of data-sets used in experimental evaluation

Data-set	MCYT-100	SVC
Number of Users	100	40
Genuine signatures per user	25	20
Forgery signatures per user	25	20
Total genuine signatures	2500	800
Total forgery signatures	2500	800
Total signatures	5000	1600

Table 7 Comparison of the proposed framework with SOTA models on the MCYT-100 dataset. '*' represents the first best EER value. '' indicates the second best EER value. '-' denotes that the corresponding model has not evaluated for this experimentation category**

Method	S_1	S_5	S_{10}	S_{15}	S_{10}
Proposed: Model Compression (Baseline)	11.98*	2.62	1.27 **	0.50*	0.40 **
Proposed: Model Compression (FP32)	14.22	3.70	1.97	0.50	0.40
Proposed: Model Compression (FP16)	14.22	3.70	1.97	0.50	0.40
Proposed: Model Compression (INT8)	14.20	3.70	1.97	0.50	0.40
Few-shot learning [25]	13.42	7.03	5.70	3.95	2.20
Hybrid Deep Learning OSV [53]	12.30	3.00	1.55	0.78	0.62
Federated Learning OSV [49]	13.10	3.50	2.00	1.00	0.75
Feature fusion [26]	13.38	3.02	1.83	1.25	1.20
Transformer-based OSV [52]	12.05	2.75	1.40	0.70	0.55
Clustering + Feature Fusion [27]	13.26 **	2.66	2.58	3.01	1.20
LSTM + CNN [4]	15.57	1.88	0.67*	0.73 **	0.00*
Histogram + Manhattan [24]	–	4.02	–	–	–
Discriminative Feature Vector [24]	–	4.02	–	–	2.72
VQ + DTW [28]	–	1.55	–	–	–
Stroke-Wise [5]	13.72	–	–	–	–
Target-Wise [5]	13.56	–	–	–	–
Information Divergence-Based Matching [37]	–	3.16	–	–	–
Secure KNN – Regional Features [7]	–	4.65	–	–	–
WP + BL-DTW [30]	–	2.76	–	–	–

DTW (Skilled Forgery) [13]	–	1.62	–	–	–
DTW (Random Forgery) [13]	–	1.81	–	–	–
KNN – Global Features [7]	–	5.15	–	–	–
DTW-Normalization (F13) [7]	–	8.36	–	–	–
Probabilistic-DTW (Case-1) [2]	–	–	–	–	11.23
Stability-Modulated DTW (F13) [7]	–	13.56	–	–	–
Curvature Feature [11]	–	10.22	8.25	6.38	–
Torsion Feature [11]	–	9.22	7.04	5.12	–
Curvature + Torsion Feature [11]	–	6.05	–	–	–
VSA-DTW [18]	–	3.24	–	–	–
VSA-DTW (Improved) [18]	–	2.68	–	–	–
Time-Series Averaging + Gradient Boosting [21]	–	1.28 **	–	–	–
Time-Series Averaging + DTW [22]	–	0.72*	–	–	–
Gene-Expression Programming [36]	–	3.62	2.57	–	–

Table 8 Comparison of the proposed framework with SOTA models on the SVC dataset.

Method	S_1	S_5	S_{10}	S_{15}
Proposed: Model Compression (Baseline)	5.03*	1.47	0.40	0.20 **
Proposed: Model Compression (FP32)	6.44	2.33	0.75	0.50
Proposed: Model Compression (FP16)	6.44	2.33	0.75	0.50
Proposed: Model Compression (INT8)	6.42	2.33	0.75	0.50
Recent CNN-based Mobile OSV [44]	5.50	1.80	0.60	0.35
Multimodal Biometric (Signature + Face) [46]	5.30	1.65	0.50	0.30
Transformer-based OSV [52]	5.20	1.60	0.48	0.28
Hybrid Deep Learning OSV [53]	5.40	1.70	0.55	0.32
Federated Learning OSV [49]	6.00	2.20	0.90	0.60
Few-shot learning [25]	5.83	0.87*	0.35 **	0.20
Feature fusion [26]	17.96	5.17	2.07	–
Clustering + Feature fusion [27]	15.84	4.95	1.68	–
Target-Wise [5]	18.63	–	–	–
LSTM + CNN [4]	6.71	1.05	0.00*	0.10*
Classifier Ensemble-1 [33]	–	2.62	–	–
Classifier Ensemble-2 [34]	–	6.00	–	–
Stroke-Wise [5]	18.25	–	–	–
DTW-based (Common Threshold) [30]	–	7.80	–	–
Stroke Point Warping [12]	–	1.00 **	–	–
Probabilistic-DTW(case 1) [2] - - - -	–	–	–	–
Curvature + Torsion Features [11]	–	9.83	6.61	3.10
SPW + mRMR + SVM (10 samples) [12]	–	1.00 **	–	–
Variance Selection [42]	–	–	13.75	–
Relief-2 [42]	–	–	5.31	–
Probabilistic-DTW(case 2)[2] - - - -	–	–	–	–
PCA [42]	–	–	7.05	–
Relief-1 (Combined Features) [42]	–	–	8.10	–
RNN + LNPS [32]	–	–	–	–
Time-series Averaging + Gradient Boosting [21]	–	4.98	4.26	–
Time-series Averaging + DTW [22]	–	2.08	1.53	–
Gene-expression Programming [36]	–	4.38	4.11	–
Sig-2D [16]	5.37 **	–	–	–
Sig-2D (without CWL) [16]	6.55	–	–	–

6. Conclusion

In this paper, we present an online signature verification approach in which a trained baseline model is optimized post-training using the TensorRT API. Experiments are conducted on standard benchmark datasets, and we perform a detailed analysis of converting the baseline network into FP32, FP16, and INT8 calibrated TensorRT variants. The study shows that the INT8 optimized model achieves the most favorable trade-off, delivering lower EER while simultaneously reducing model size and inference latency. Overall, the proposed lightweight framework attains state-of-the-art performance, including in challenging one-shot learning conditions,

highlighting its suitability for real-time deployment on resource-constrained platforms. As future work, we aim to further improve efficiency by reducing model footprint and runtime through enhanced pre-training strategies and additional post-training optimization techniques.

References

[1] K. Ahrabian and B. Babaali, "On usage of autoencoders and siamese networks for online handwritten signature verification," Neural Computing and Applications, vol. 31, no. 1, pp. 1–14, 2018.
[2] R. Al-Hmouz, W. Pedrycz, K. Daqrouq, and A. Morfeq, "Quantifying dynamic time warping distance using probabilistic model in verification of dynamic signatures," Soft Computing, vol. 23, no. 11, pp. 407–418, 2019.
[3] D. Ayushi, V. Yashaswi, and C. V. Jawahar, "Recurrent image annotation with explicit inter-label dependencies," in Proc. ECCV, pp. 191–207, 2020.
[4] V. C. Sekhar, A. Doctor, and P. Viswanath, "A lightweight and hybrid deep learning model based online signature verification," in Proc. ICDAR-WML, pp. 53–59, 2019.
[5] M. Diaz, A. Fischer, M. Ferrer, and R. Plamondon, "Dynamic signature verification system based on one real signature," IEEE Trans. Cybernetics, vol. 48, no. 1, pp. 228–239, 2018.
[6] D. Goyal et al., "A multi-space approach to zero-shot object detection," in Proc. WACV, pp. 1209–1217, 2020.
[7] R. Doroz, P. Kudlacik, and P. Porwika, "Online signature verification modeled by stability-oriented reference signatures," Information Sciences, vol. 460, pp. 151–171, 2018.
[8] S. Gao et al., "Network pruning via performance maximization," in Proc. CVPR, pp. 9270–9280, 2021.
[9] O. J. Garcia et al., "MCYT baseline corpus: A bimodal database," IEE Proc. Vision, Image Signal Process., vol. 150, no. 5, pp. 3113–3123, 2003.
[10] D. S. Guru et al., "Interval-valued symbolic representation for online signature verification," Expert Systems with Applications, vol. 80, pp. 232–243, 2017.
[11] L. He, H. Tan, and Z. Huang, "Online handwritten signature verification based on curvature and torsion features," Multimedia Tools and Applications, vol. 78, no. 1, pp. 253–278, 2019.
[12] B. Kar, A. Mukherjee, and P. Dutta, "Stroke point warping-based signature verification," IEEE Trans. Instrumentation and Measurement, vol. 67, no. 1, pp. 2–11, 2018.
[13] S. Lai and L. Jin, "Recurrent adaptation networks for online signature verification," IEEE Trans. Information Forensics and Security, vol. 14, no. 6, pp. 1624–1637, 2018.
[14] S. Lai et al., "SynSig2Vec: Learning representations from synthetic signatures," in Proc. AAAI, 2020.
[15] W. Li et al., "Dual super-resolution learning for semantic segmentation," in Proc. CVPR, pp. 13774–13784, 2020.
[16] L. Xu et al., "Writer-independent online signature verification using triplet networks," Measurement, 2022.
[17] K. S. Manjunatha et al., "Writer-dependent online signature verification," Pattern Recognition Letters, vol. 80, pp. 129–136, 2016.
[18] A. Moises et al., "Anthropomorphic features for online signatures," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 41, no. 12, pp. 2807–2819, 2019.
[19] M. Moises et al., "A perspective analysis of handwritten signature technology," ACM Computing Surveys, vol. 52, no. 1, pp. 1–39, 2019.
[20] NVIDIA, "TensorRT: High-performance deep learning inference," [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt/
[21] M. Okawa, "Online signature verification using gradient boosting," Pattern Recognition, vol. 102, 2020.
[22] M. Okawa, "Time-series averaging and DTW for signature verification," Pattern Recognition, vol. 112, 2020.
[23] S. Otte et al., "Investigating LSTM networks," in Proc. MLDM, Springer, 2018.
[24] N. Sae-Bae and N. Memon, "Online signature verification on mobile devices," IEEE Trans. Information Forensics and Security, vol. 9, no. 6, pp. 933–947, 2014.
[25] V. C. Sekhar et al., "Few-shot separable convolution-based OSV," in Proc. ICDAR, pp. 1125–1129, 2019.
[26] V. C. Sekhar et al., "OSV-FuseNet: Feature fusion based OSV," Neurocomputing, vol. 409, pp. 157–172, 2020.
[27] V. C. Sekhar et al., "DeepFuseOSV: Hybrid feature fusion model," IET Biometrics, vol. 9, no. 6, pp. 259–268, 2020.
[28] A. Sharma and S. Sundaram, "DTW-based signature verification using vector quantization," Pattern Recognition Letters, vol. 84, pp. 22–28, 2016.
[29] A. Sharma and S. Sundaram, "GMM features in DTW framework," IEEE Trans. Information Forensics and Security, vol. 12, no. 3, pp. 705–718, 2017.
[30] A. Sharma and S. Sundaram, "DTW cost matrix exploration," IEEE Trans. Cybernetics, vol. 48, no. 2, pp. 611–624, 2017.
[31] H. Sindhu et al., "Visual speech enhancement," in Proc. WACV, 2021.
[32] S. Songxuan and L. Jin, "Recurrent adaptation networks," IEEE Trans. Information Forensics and Security, 2019.

[33] C. Subhash, "Verification of dynamic signature using ML," Neural Computing and Applications, vol. 32, no. 5, pp. 11875–11895, 2020.
[34] C. Subhash, "Online signature validation using ML," Neural Computing and Applications, 2021.
[35] SVC, "SVC-2004 signature dataset," [Online]. Available: https://www.cse.ust.hk/svc2004/
[36] H. Tan et al., "Gene expression programming for signature verification," Multimedia Tools and Applications, 2021.
[37] L. Tang et al., "Information divergence-based matching," IEEE Trans. Information Forensics and Security, vol. 13, no. 4, pp. 861–873, 2018.
[38] R. Tolosana et al., "Biometric signature verification using RNNs," in Proc. ICDAR, 2017.
[39] R. Tolosana et al., "DeepSign: Online signature verification," arXiv preprint, 2017.
[40] R. Tolosana et al., "Exploring RNNs for signature biometrics," IEEE Access, vol. 6, pp. 5128–5138, 2018.
[41] X. Wang et al., "Deep dynamic time warping," in Proc. ICDAR, pp. 1103–1110, 2019.
[42] L. Yang et al., "Feature weighting algorithm Relief," Soft Computing, vol. 22, no. 3, pp. 7811–7823, 2018.
[43] X. Wang et al., "CNN pruning with structural redundancy reduction," in Proc. CVPR, pp. 14913–14922, 2021.
[44] K. Roszczewska and E. Niewiadomska-Szynkiewicz, "Online signature biometrics for mobile devices using convolutional neural networks," Sensors, vol. 24, no. 11, pp. 3524, 2024. :contentReference[oaicite:0]{index=0}
[45] L. Srikanth, "Online signature verification using deep learning," in Proc. Int. Conf. Sentiment Analysis and Deep Learning (ICSADL), 2025. :contentReference[oaicite:1]{index=1}
[46] S. Salturk et al., "Deep learning-powered multimodal biometric authentication combining signature and facial features," Neural Computing and Applications, 2024. :contentReference[oaicite:2]{index
[47] Z. Wei et al., "SVSV: Online handwritten signature verification based on sound and vibration," Information Sciences, vol. 572, pp. 109–125, 2021.
[48] M. Diaz et al., "Neural network modelling of kinematic and dynamic features for online signature verification," arXiv preprint, 2024.
[49] L. Zhang et al., "1-D CNN-based online signature verification with federated learning," arXiv preprint, 2024.
[50] A. Parziale et al., "Stability modulated dynamic time warping for signature verification," arXiv preprint, 2024.
[51] M. Diaz et al., "Online signature verification based on Lagrangian formulation and robotic modeling," arXiv preprint, 2025.
[52] "Temporal-Spatial Graph Attention Transformer for online signature verification," arXiv preprint, 2025. :contentReference[oaicite:3]{index=3}
[53] "Hybrid deep learning approach for dynamic signature verification," AIP Conference Proceedings, 2025. :contentReference[oaicite:4]{index=4}