Unsupervised Sensor-based Continuous Authentication with Low-Rank Transformer Using Learning-to-Rank Algorithms

Zhenyu Yang, Yantao Li, *Senior Member,* and Gang Zhou, *Fellow, IEEE*

***Abstract*—With the rapid development of the Internet of Things (IoTs) and mobile communications, mobile devices have become indispensable in our daily lives. Given the substantial amount of private information stored on these devices, the security of mobile devices has emerged as a significant concern for users. Different from conventional methods such as PINs, fingerprints, and face IDs, which authenticate users only during the initial login stage, continuous authentication ensures consistent verification while mobile devices are in use. Current continuous authentication methods require extensive data from a series of users for effective training. Nevertheless, it is challenging to collect sufficient amount of data within a limited time. In this paper, we propose CALL, an unsupervised sensor-based Continuous Authentication system with a Low-rank transformer using Learning-to-rank algorithms. The lightweight CALL is capable of providing both spatial and temporal features for end-to-end authentication. Specifically, CALL utilizes time series data from a legitimate user, collected by the accelerometer, gyroscope, and magnetometer sensors on smartphones, to train a pure one-dimensional autoencoder for spatial features and a shuffle low-rank Transformer (SLRT) for temporal features in the training phase. In the authentication phase, the trained pure one-dimensional autoencoder captures spatial features by reconstructing input data to obtain the reconstruction error, and SLRT captures temporal features by predicting a ranking vector that reveals the order of the shuffled feature sequence. The predicted ranking vector is then used to recover the shuffled sequence and the similarity between the frequency spectrum sequences of the recovered sequence and the original time series data is calculated. The reconstruction error and similarity are compared against pre-defined thresholds, and CALL authenticates a user as legitimate only if both values fall below their respective thresholds. Finally, we evaluate the performance of CALL on UCI_HAR, WISDM_HARB, and our dataset, and the extensive experiments illustrate that CALL reaches the best performance with 96.43%, 95.24% and 96.92% accuracy, and 4.28%, 4.76% and 3.86% EERs on the three datasets, outperforming state-of-the-art continuous authentication methods.**

***Index Terms*—Continuous authentication, Low-rank Transformer, Learning-to-rank algorithms, Temporal features, Spatial features.**

I. Introduction

NOWADAYS, mobile devices, such as smartphones and smartwatches, have become indispensable tools in our daily lives. They provide significant convenience for people to conduct electronic payments using digital wallets, share documents through file transfer applications, and communicate via chat software. As a result, these devices store a substantial amount of private and sensitive information, such as bank account details, confidential files, chat records, and personal photos. To protect against information disclosure, it is imperative to implement reliable authentication mechanisms to ensure the security of mobile devices. Common authentication mechanisms include the use of PINs, graphical passwords, Face IDs, and fingerprints to unlock mobile devices. However, these methods are one-time authentication solutions, offering protection solely during the login period. Studies indicate that smudge attacks [1], shoulder surfing attacks [2], and password inference attacks [3] can potentially compromise these one-time authentication mechanisms. Moreover, these methods do not detect illegal users once devices are successfully logged in.

To address these issues, continuous authentication mechanisms are designed to provide constant security protection for the duration of device usage. Research on sensor-based human activity recognition [4, 5, 6, 7] has inspired the development of behavioral data-based continuous authentication, which can be regarded as unique user signatures. Various types of continuous authentication systems leverage different biometric behavioral data, including walking gait [8, 9], gestures [10, 11], keystrokes [12, 13], and touchscreen dynamics [14, 15], to extract the behavioral patterns through device usage [16, 17]. These continuous authentication systems successfully authenticate users even under forgery attacks [18]. With the rapid development of deep learning, continuous authentication models based on deep neural networks (DNNs) can capture representative temporal features of behavioral biometrics [19]. To train such DNNs for continuous authentication systems, a substantial amount of data from diverse users are required, leading to the exploration of data augmentation methods and models [20, 21, 22] for addition data generation. Then, these collected and augmented data undergo cleaning and denoising, labeled corresponding classes for different users. Finally, the processed and labeled data are used to train the designed DNNs to learn representative features. Based on our knowledge, there are several challenges for DNN-based continuous authentication models:

Zhenyu Yang and Yantao Li are with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: {20172725,yantaoli}@cqu.edu.cn).

Gang Zhou is with the Department of Computer Science, William & Mary, Williamsburg, VA, 23185, USA (e-mail: gzhou@cs.wm.edu).

Manuscript received September XX, 2023; revised XXXX XX, 201X. This work was supported in part by the National Natural Science Foundation of China under Grants 62072061 and U20A20176. (Corresponding author: Yantao Li.)

- To collect substantial and diverse data, several subjects or volunteers (10 or more) are required to participate. However, it is challenge to ask volunteers to perform repetitive actions on the specified mobile devices for long periods, resulting in limited data collected from volunteers.
- It is difficult to determine whether the labels of data change after augmentation. While various studies propose many data augmentation methods for continuous authentication systems, there is a lack of regulations to ascertain whether the generated sensor-based data maintain the labels of original data, especially when the original data are insufficient.
- DNN modules pre-trained on extensive datasets comprise multiple layers and excessive parameters, posing the challenge of their implementation on mobile devices with limited computation and storage resources.
- Existing unsupervised DNN modules based on autoencoders solely provide a single reconstruction error as the metric for authenticating users, leading to poor performance [23].

To overcome these challenges, we propose CALL, a novel unsupervised sensor-based Continuous Authentication system with a Low-rank Transformer using Learning-to-rank algorithms, which serves as a lightweight and end-to-end authentication mechanism. Different from methods that solely use one model constructed from CNNs or recurrent neural networks (RNNs) to capture complex spatial and temporal features [24, 25, 26], CALL captures these two types of features through two independent modules, respectively. Specifically, we design an autoencoder based on a pure one-dimensional CNN (pure-CNN) to capture spatial features of sensor-based time series data and calculate a reconstruction error exclusively using legitimate user data during the registration phase. The reconstruction error is subsequently used as one factor to authenticate the users' identities. Simultaneously, the designed shuffled low-rank Transformer (SLRT) encoder captures temporal features of data in a low-dimension feature space by: 1) predicting the ranking vector of randomly shuffled time series data processed by the encoder of the pure CNN autoencoder; 2) recovering the shuffled data according to the predicted ranking vector; and 3) calculating the similarity between frequency spectrum sequences of the original data and the recovered time series data using dynamic frequency wrapping (DFW) algorithms. The user is identified as legitimate when both the similarity and RE fall below their respective thresholds.

Thus, our contributions can be summarized as follows:

- We propose CALL, a lightweight and end-to-end unsupervised Continuous Authentication system based on a Low-rank transformer using Learning-to-rank algorithms. Based on biometric behavioral data collected by smartphone sensors, CALL employs the designed pure-CNN autoencoder and SLRT, trained solely on the legitimate user data, to provide both spatial and temporal features for end-to-end authentication.
- We design an autoencoder based on a pure one-dimensional CNN to extract spatial features of raw sensor data. The encoder in the autoencoder maps the data into a low-dimension feature space, thereby effectively reducing the parameters of downstream modules.
- We design SLRT based on the low-rank Transformer encoder to extract temporal features by predicting the ranking of the feature sequence from the encoder of the autoencoder. We are among the first to utilize the lightweight SLRT and learning-to-rank algorithms for achieving unsupervised training in continuous authentication systems.
- The experimental results demonstrate that CALL outperforms state-of-the-art models in both accuracy and EERs on UCI_HAR (96.43% and 4.28%), WISDM_HARB (95.24% and 4.76%) and our dataset (96.92% and 3.86%), respectively.

The remainder of this paper is organized as follows: Section II reviews the representative deep-learning continuous authentication systems. Section III presents the preliminary knowledge of DTW and Transformer for CALL. Section IV details the proposed CALL in terms of overview, pure-CNN autoencoder, SLRT, training policy, and authentication. Section V evaluates the performance of CALL through comparisons with state-of-the-art systems and ablation studies. Section VI concludes this work and discusses the future work.

II. Related Work

Commonly used authentication methods on mobile devices are typically one-time identification mechanisms, including PINs, fingerprints, face IDs, and graphical patterns. These methods solely identify users during the login process, leaving unlocked devices vulnerable to potential imposters. Deep-learning continuous authentication methods based on behavioral biometrics provide a consistent and robust security solution for mobile devices. In particular, continuous authentication can effectively prevent forgery attacks, where imposters attempt to gain access to mobile devices through random or skillful means [18]. These methods are traditionally categorized based on different behavioral modalities, such as voice-based, motion-based, gestures-based, keystroke dynamics-based, gait-based, and hand waving-based continuous authentication mechanisms [27, 28, 29, 30]. In this section, we categorize existing deep-learning continuous authentication systems based on different training policies into three groups: pre-trained models, supervised-trained models, and unsupervised-trained models.

A. Pre-trained Models

The pre-trained model typically comprises a pre-trained feature extraction network model and a one-class classification (OCC) module. Specifically, the feature extraction module is pre-trained on a large dataset in a supervised manner to learn the expression capability. In the registration phase, the data of the legitimate user are processed by the pre-trained feature extraction module and are then used to train the OCC module. In the authentication phase, all user features are extracted and then discriminated by the trained OCC. The authors of [22] propose a feature fusion method to strengthen the designed

CNN-based feature extraction module and use the one-class supporting vector machine (OC-SVM) as the OCC module. The behavioral data are transformed into the frequency domain by Fourier transformation and fused with the data in the time domain after two pre-trained feature extraction modules. The fused feature vector is proved to be more discriminative, and the OC-SVM used in the OCC module exhibits the best performance compared with various machine learning OCC methods, such as k-nearest neighbors (kNN), random forest (RF) and decision tree (DT). The authors in [31] propose a novel convolution-based Transformer model, which combines CNN and a self-attention mechanism to extract high-level contextual semantic information from original time series data. Moreover, this feature extraction model is pre-trained on one dataset but demonstrates generalizability to achieve continuous authentication with OC-SVM on another dataset.

Pre-trained models, which encompass pre-trained and OCC modules, encounter challenges in achieving end-to-end training, leading to intricate training processes. Moreover, they demand a substantial volume and diverse classes of data for pre-training, increasing the cost of sufficient data collection. In contrast, CALL undergoes training solely on the data from the legitimate user, achieving end-to-end training with simplicity and efficiency.

B. Supervised-trained Models

The supervised-trained model is a binary classifier trained on an extensive dataset comprising two classes: positive and negative. The positive data are from genuine users during the registration phase, while the negative data are from imposters during the authentication phase. The trained model is then employed for authentication by discriminating whether the user falls into the positive or negative class. In [32], the authors put forward models based on three long short-term memory (LSTM) backbones: simple LSTM, bidirectional LSTM, and multilayer LSTM. After training these models on the same dataset, the bidirectional LSTM-based model exhibits the best average F_1_Score. The authors of [23] adopt three data augmentation methods, i.e., jittering, permutation, and scaling, to increase dataset volume. A multilayer LSTM encoder is designed for binary classification on each augmented dataset for authentication. In [24], the authors combine the one-dimensional residual network (1D-ResNet) and the squeeze-and-excitation to create the pure CNN-based feature extraction module, named 1D-ResNet-SE, for capturing patterns in time series data. The downstream fully connected layers process binary classification with the extracted features, and the results demonstrate that the proposed model performs better than models based on simple CNN and LSTM.

Supervised-trained models can capture users' patterns from the training set and achieve end-to-end training. However, they face challenges when generalizing to real-world scenarios with many unseen users and limited legitimate data. In contrast, CALL demonstrates high performance in the authentication phase, effectively accommodating both seen and unseen users.

C. Unsupervised-trained Models

The unsupervised-trained model captures features using only one legitimate user without any given label during the training stage and can distinguish the imposters and the legitimate users at the authentication stage. To the best of our knowledge, studies on unsupervised-trained models for continuous authentication systems are relatively rare. The authors of [25] propose an unsupervised LSTM-based autoencoder to reconstruct the input data during the training phase. During the authentication phase, the trained autoencoder is used to recover the test data, and then the reconstruction error is calculated. If the error exceeds the threshold, the user is regarded as an imposter; otherwise, as the legitimate user. They demonstrate that the designed unsupervised LSTM-based autoencoder performs worse than the supervised binary classifier on many occasions. In [33], the authors propose a CNN-based autoencoder and an LSTM-based module to achieve unsupervised multi-task training. Specifically, the autoencoder reconstructs the input time series data and minimizes the reconstruction error, while the LSTM-based module takes the reconstructed data as input and recognizes the specific gait. Besides, the authors design a style loss to ensure that the reconstructed data do not contain private information other than authentication.

Unsupervised-trained models are only trained on data from the legitimate user, capturing the pattern of this specific user, which allows for end-to-end training and requires a minimal volume of data. Therefore, deep-learning unsupervised-trained models are promising, but their exploration has been limited. Existing studies mainly focus on autoencoders, where spatial and temporal features are processed together, resulting in suboptimal accuracy [25, 33]. In this work, CALL, our proposed model, represents a new unsupervised-trained approach utilizing both spatial and temporal features for authentication. To the best of our knowledge, it is the first deep-learning continuous authentication model based on autoencoder and low-rank Transformer, trained through learning-to-rank algorithms.

III. Preliminary

CALL incorporates a specially designed low-rank Transformer (LRT) module, the low-rank version of the classic Transformer encoder architecture. Besides, inspired by the time warping of dynamic time warping (DTW) [34], we design the dynamic frequency wrapping (DFW) algorithm for calculating the similarity between two frequency spectrum sequences. To better understand CALL, we provide preliminary knowledge on the Transformer encoder and DTW algorithm in this section.

A. Transformer Encoder

The Transformer architecture, comprising an encoder and a decoder, is a prominent sequence-to-sequence model widely used in machine translation [35]. Besides, the pre-trained encoder of the Transformer serves as the backbone for various classification models, including bidirectional encoder representations from Transformers (BERT) [36] and cross-lingual language models (XLMs) [37]. In this section, we provide a preliminary introduction to the Transformer encoder.

The classic Transformer encoder comprises two main modules: an attention module and a positional feed-forward network (FFN) module. In the attention module, three matrices $\boldsymbol{Q} \subseteq \mathbb{R}^{N \times D_k}$, $\boldsymbol{K} \subseteq \mathbb{R}^{M \times D_k}$, and $\boldsymbol{V} \subseteq \mathbb{R}^{M \times D_v}$ are used to compute attention through the scaled dot-product operation in Eq. (1):

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D_k}})\boldsymbol{V}, \quad (1)$$

where N and M denote the lengths (or values) of queries and keys, D_v and D_k are dimensions of queries and keys. $\sqrt{D_k}$ is divided by the product of $\boldsymbol{Q}$ and $\boldsymbol{K}$, which aims to avoid gradient vanishing. The scaled dot-product in Transformer is applied in a multi-head attention manner, where the original sequence with D_m dimensions is first projected into $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ using three learnable matrices, and then segmented into g different heads for each of them. In each head, the dimensions of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ become $\frac{D_k}{g}$, $\frac{D_k}{g}$ and $\frac{D_v}{g}$, respectively, and the attention operation in each head is performed as in Eq. (1). Finally, the multi-head attention module concatenates the outputs of all the g heads to form a final self-attention vector $\boldsymbol{A}$ with the same shape as the input data. Transformer incorporates a residual connection between $\boldsymbol{A}$ and the input data, which are further processed by the FFN. The FFN module is a fully connected layer that operates as a non-linear activation projection of the outputs from the attention module, enhancing the expressive capability of the Transformer.

In this work, we enhance the classic Transformer encoder to create a low-rank framework for capturing temporal features in time series data. The improved Transformer features fewer parameters and reduced inference latency while maintaining outstanding performance.

B. DTW

For two similar yet misaligned time series, DTW can yield a relatively small distance between them. In contrast, the Euclidean distance, which assesses the point-to-point similarity between two time series, is large. Fig. 1(a) illustrates an example where DTW calculates the distance between two misaligned yet similar sequences $\boldsymbol{p} = [p_1, \cdots, p_n]$ and $\boldsymbol{q} = [q_1, \cdots, q_m]$, showing a delay between them. This is more reasonable than the Euclidean distance as depicted in Fig. 1(b). In Fig. 1, each dashed line represents the distance between two corresponding connected points. Therefore, two identical time series with minimal latency will exhibit a small DTW distance value. Assuming $\boldsymbol{p}$ is a template sequence, classic DTW is implemented through dynamic programming (DP): First, a $n \times m$ matrix $\boldsymbol{D}$ is constructed, where $d_{i,j}$ denotes the Euclidean distance between p_i and q_j. Then, the accumulated distance matrix $\boldsymbol{A}$ based on $\boldsymbol{D}$ can be calculated by Eq. (2):

$$a_{i,j} = min[a_{i-1,j-1}, a_{i-1,j}, a_{i,j-1}] + d_{i,j}, \quad (2)$$

Finally, the minimum distance path l from $a_{n,m}$ to $a_{1,1}$ in $\boldsymbol{A}$ can be searched by the DP algorithm, and each node $a_{k,o}(k, o \in l)$ in this path represents the correspondence between p_k and q_o. The optimization of DTW algorithms, as outlined by Sakoe and Chiba [34], involves three main conditions:

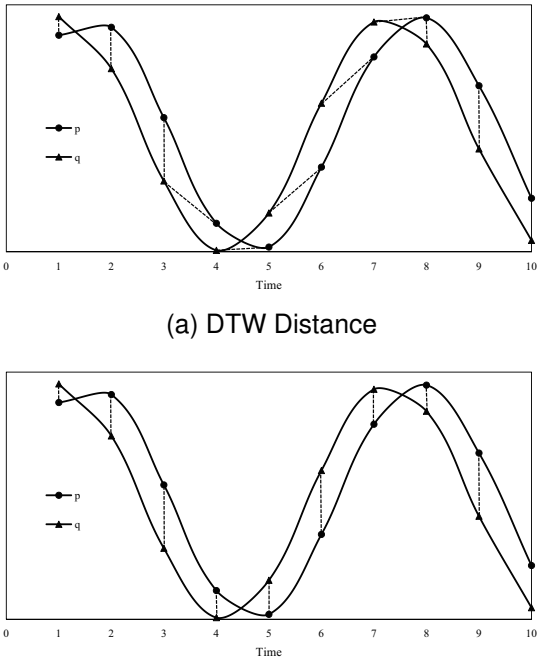


Fig. 1. DTW and Euclidean distance between p and q. Sequences $\boldsymbol{p}$ and $\boldsymbol{q}$ are similar but there is a delay between them, and dashed lines illustrate the matching mode when calculating the distance.

Boundary Condition: The starting and ending points are at the bottom-left corner and the top-right corner of $\boldsymbol{A}$, respectively.

Continuity Condition: The path l moves one step at a time. Specifically, both k and o in l can only increase by one on each step along the path.

Monotonicity Condition: The points on the path l monotonically progress over time and cannot retreat to the left or downside.

Our DFW employs the same algorithm as DTW. However, DFW compares the similarity of two frequency spectrum sequences, while classic DTW calculates distances between two time sequences.

IV. Methods

We propose CALL, an unsupervised sensor-based continuous authentication system using a low-rank Transformer and learning-to-rank algorithms. CALL ensures continuous authentication by using a pure-CNN autoencoder module and an SLRT module to extract spatial and temporal features, respectively. Specifically, in the unsupervised training phase, both modules are trained using the enrollment behavioral data of a legitimate user. In the authentication phase, CALL labels the behavioral features of the current user as either "a legitimate user" or "an imposter" by leveraging the output of the trained pure-CNN autoencoder and SLRT, combined with designed authentication algorithms. In this section, we first present an overview of CALL and then detail the autoencoder and SLRT modules. Next, we introduce formalized definitions

TABLE I
SUMMARY OF TERMS AND NOTATIONS.

	Symbol	Definition
Training Phase	$\boldsymbol{x} \subseteq \mathbb{R}^{T\times D} = [\boldsymbol{x^1}\cdots\boldsymbol{x^T}]^\top$	Original time series data
	$\hat{\boldsymbol{x}} \subseteq \mathbb{R}^{T\times D} = [\hat{\boldsymbol{x}}^1\cdots\hat{\boldsymbol{x}}^T]^\top$	Reconstructed time series data from the decoder
	$\boldsymbol{x_{ex}} \subseteq \mathbb{R}^{T\times D_{ex}} = [\boldsymbol{x^1_{ex}}\cdots\boldsymbol{x^T_{ex}}]^\top$	Spatially extended time series data
	$\boldsymbol{f} \subseteq \mathbb{R}^{T\times D_e} = [\boldsymbol{f^1}\cdots\boldsymbol{f^T}]^\top$	Sequential feature from the encoder
	$\boldsymbol{O} \subseteq \mathbb{N}^T = [o_1\cdots o_T]^\top (o_i = i)$	Real order part of $\boldsymbol{f}$ and $\boldsymbol{x}$
	$\boldsymbol{f_s} \subseteq \mathbb{R}^{T\times D_e} = [\boldsymbol{f^{\sigma(1)}}\cdots\boldsymbol{f^{\sigma(T)}}]^\top$	Shuffled sequential feature of $\boldsymbol{f}$
	$\boldsymbol{O_s} \subseteq \mathbb{N}^T = [o_{\sigma(1)}\cdots o_{\sigma(T)}]^\top$	Shuffled order part of $\boldsymbol{f_s}$ based on $\boldsymbol{O}$
	$\boldsymbol{R} \subseteq \mathbb{R}^T = [r_1\cdots r_T]^\top$	Ranking vector output by SLRT
Authentication Phase	$\boldsymbol{x_{test}} \subseteq \mathbb{R}^{T\times D} = [\boldsymbol{x^1_{test}}\cdots\boldsymbol{x^T_{test}}]^\top$	Test original time series data
	$\hat{\boldsymbol{x}}_{\boldsymbol{test}} \subseteq \mathbb{R}^{T\times D} = [\hat{\boldsymbol{x}}^1_{\boldsymbol{test}}\cdots\hat{\boldsymbol{x}}^T_{\boldsymbol{test}}]^\top$	Reconstructed test time series data from the decoder
	$\boldsymbol{x_s}_{\boldsymbol{test}} \subseteq \mathbb{R}^{T\times D} = [\boldsymbol{x^1_s}_{\boldsymbol{test}}\cdots\boldsymbol{x^T_s}_{\boldsymbol{test}}]^\top$	Shuffled test original time series data
	$\boldsymbol{f_s}_{\boldsymbol{test}} \subseteq \mathbb{R}^{T\times D_e} = [\boldsymbol{f^{\sigma(1)}_{test}}\cdots\boldsymbol{f^{\sigma(T)}_{test}}]^\top$	Shuffled sequential feature of $\boldsymbol{f_{test}}$
	$\boldsymbol{R}_{\boldsymbol{test}} \subseteq \mathbb{R}^T = [\boldsymbol{r}_{\boldsymbol{test_1}}\cdots\boldsymbol{r}_{\boldsymbol{test_T}}]^\top$	Ranking vector output by SLRT
	$\boldsymbol{x}_{\boldsymbol{test\_r}} \subseteq \mathbb{R}^{T\times D} = [\boldsymbol{x}_{\boldsymbol{test\_r}_1}\cdots\boldsymbol{x}_{\boldsymbol{test\_r}_T}]^\top$	Recovered test original time series data based on $\boldsymbol{x_s}_{\boldsymbol{test}}$ and $\boldsymbol{R}_{\boldsymbol{test}}$
	$\boldsymbol{x^{iT}_{test_i}} \subseteq \mathbb{R}^T = [x_{test_1}\cdots x_{test_i}]^\top$	The ith spatial dimension's sequence of $\boldsymbol{x_{test}}$
	$\boldsymbol{A_i} \subseteq \mathbb{R}^K = [A_i(\omega_1)\cdots A_i(\omega_K)]^\top$	Amplitude-frequency spectrum vector of $\boldsymbol{x^{iT}_{test_i}}$

of the proposed learning-to-rank training strategy for achieving unsupervised learning. Finally, we provide details of the proposed CAll containing sequence reorder (SR), DFW, and reconstruction error calculation. Table I summarizes the terms and notations used in this work.

A. Overview

As illustrated in Fig. 2, CALL comprises two phases: the training phase and the authentication phase. The training phase includes data collection, an autoencoder, and an SLRT. In data collection, sensors embedded in mobile device, such as the accelerator, gyroscope, and magnetometer, collect time series data when the legitimate owner uses the device, which are related to the behavioral patterns of the owner. The collected raw sensor data within a specific period undergo preprocessing, including denoising and normalization and are then stored as the owner's profile. This profile is used for training the downstream autoencoder and SLRT. The autoencoder module consists of an encoder and a decoder, both implemented as pure 1D-convolution networks. This module focuses on capturing spatial features of sensor-based time series data. The encoder employs hierarchical 1D-convolution layers with nonlinear activation functions to map time series data $\boldsymbol{x}$ into a feature sequence $\boldsymbol{f}$, which maintains the time causality and has the same length as $\boldsymbol{x}$ but with lower dimensions. The decoder, also using hierarchical 1D-convolution structures, reconstructs $\hat{\boldsymbol{x}}$ from $\boldsymbol{f}$. The optimization objective of the autoencoder is to minimize the distance between $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}$. Thus, during training, the autoencoder captures spatial features of $\boldsymbol{x}$ through a series of 1D convolution operations.

Since $\boldsymbol{f}$ serves as the input for the SLRT module, the proposed SLRT operates in the feature space, aiming to capture temporal features of the time series. The SLRT module comprises two parts: the sequence shuffle module and the low-rank Transformer. After the pure 1D-convolution encoder, the feature sequence $\boldsymbol{f}$ can also be considered as time series data since the time causality is preserved in $\boldsymbol{f}$. In the SLRT module, the sequence shuffle module first disrupts the order of time steps of $\boldsymbol{f}$ to obtain an unordered sequence $\boldsymbol{f_s}$. Next, the low-rank Transformer predicts a ranking vector using $\boldsymbol{f_s}$ as the input. Specifically, the ith element in the ranking vector predicts the information of the actual order of $\boldsymbol{f_s^i}$. Thus, in the training stage, the objective of the autoencoder is to minimize the distance between $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}$, while SLRT aims to minimize the distance between the predicted ranking vector and the actual ranking vector. During the entire training process, the autoencoder and SLRT are collaboratively optimized, representing an unsupervised multi-task optimization. Due to $\boldsymbol{f}$ with lower spatial dimensions in the feature space and the low-rank architecture of the Transformer, the computational complexity of the Transformer is reduced, significantly decreasing the chance of overfitting by reducing parameters.

In the authentication phase, sensors embedded in mobile devices collect real-time data from users, including both the legitimate user and imposters, presenting new data to CALL. The collected test data are preprocessed with the same methods as used in training. With the normalized data, the trained autoencoder reconstructs the data and the reconstruction error L_{rec} is obtained by measuring the distance between the input and output of the autoencoder. In the trained SLRT, the output feature sequence of the encoder $\boldsymbol{f_{test}}$ is taken as the input, and the sequence shuffle module disrupts the order of $\boldsymbol{f_{test}}$ to generate a disordered sequence, serving as the input of the LRT module. Note that the shuffle information SI, representing the details of the replacement involved in the shuffle operation, is recorded. Following this, the original test data are shuffled using the same replacement method as applied to $\boldsymbol{f_{test}}$ based on SI, and the shuffled original data can be reordered by the SR algorithm, using the ranking vector output by the LRT module. Finally, the FFT is applied to both the original data and the reordered data. These transformed data are utilized to calculate the frequency-spectrum similarity L_{sim} by the DFW algorithm. CALL makes decisions as: 1) the test data are from an imposter if $L_{rec}{>}Th_{rec}$ (a threshold) or $L_{sim}{>}Th_{sim}$ (a threshold); and 2) $\boldsymbol{x_{test}}$ is from the legitimate user if $L_{sim} \leq Th_{sim}$ and $L_{sim} \leq Th_{sim}$. These decisions rely on the premise that the autoencoder and SLRT perform effectively only when tested on the distribution of the target class (the class for training).

B. Autoencoder Networks

Autoencoder networks comprise an encoder and a decoder, both designed as pure hierarchical 1D-convolution networks, as shown in Fig. 3. Given the time series data $\boldsymbol{x} \subseteq \mathbb{R}^{T\times D}$, as illustrated in the upper portion of Fig. 3, the encoder initiates a 1D convolution on $\boldsymbol{x}$ to expand spatial dimensions, resulting in $\boldsymbol{x_{ex}} \subseteq \mathbb{R}^{T\times D_{ex}}$, where D_{ex} is larger than D. Then, the following 1D-convolution layers in the encoder compress the spatial dimensions of $\boldsymbol{x_{ex}}$ into a feature sequence $\boldsymbol{f} \subseteq \mathbb{R}^{T\times D_e}$. Note that the kernels of the ith 1D-convolution layer are half the size of those in the $(i-1)$th layer within the encoder. For the decoder, as illustrated in the lower portion of Fig. 3, it uses 1D-convolution layers symmetrically arranged with the encoder to reconstruct the feature sequence $\boldsymbol{f}$ to $\hat{\boldsymbol{x}}$.

When padding is applied to $\boldsymbol{x}$, the 1D-convolution kernel $\boldsymbol{k}$ processes $\boldsymbol{x}$ to calculate features by Eq. (3):

$$\boldsymbol{FEATURE_i} = \begin{cases} 0 \times \boldsymbol{k^1} + \langle \boldsymbol{x^1}, \boldsymbol{k^2} \rangle, & i = 1, \\ \sum_{j=1}^{s} \langle \boldsymbol{x^{i-s+j}}, \boldsymbol{k^j} \rangle, & i \in (1, T], \end{cases} \tag{3}$$

where s denotes the kernel size (set to 2). Features $[\boldsymbol{FEATURE_1}\cdots\boldsymbol{FEATURE_T}]$ obtained by one kernel

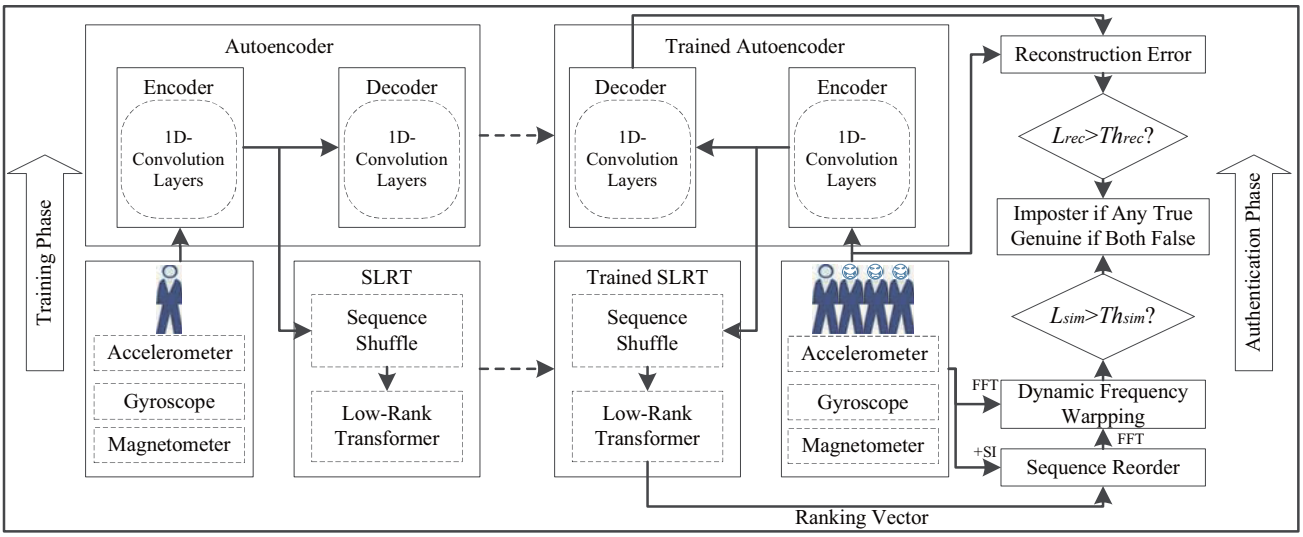


Fig. 2. Architecture of CALL. The left part represents the training phase, consisting of the training of the autoencoder and LRT using sensor-based data from a legitimate user, which are collaboratively optimized. The right part indicates the authentication phase, which uses the trained autoencoder and LRT, along with SR, DFW, reconstruction erro calculation, and FFT algorithms, to obtain reconstruction error L_{rec} and frequency spectrum similarity L_{sim} for comparison with thresholds Th_{rec} and Th_{sim}.

maintain temporal causality since the 1D-convolution kernel slides across the time axis of $\boldsymbol{x}$, and $\boldsymbol{FEATURE_i}$ corresponds to the ith sliding step, as expressed by Eq. (3). Thus, 1D-convolution layers alter only the spatial dimensions (based on the number of kernels) of $\boldsymbol{x}$, while maintaining temporal causality. The low-spatial-dimension feature sequence $\boldsymbol{f} \subseteq \mathbb{R}^{T\times D_e}$, derived through a series of 1D-convolution operations, can thus be regarded as time series data with the same temporal causality as $\boldsymbol{x}$.

C. Sequence Shuffle and Low-Rank Transformer

In this section, we introduce the SLRT module, which processes $\boldsymbol{f}$ in the low-spatial-dimension space, consequently reducing the input space of LRT.

Definition 1. For the time series data $\boldsymbol{x}$, we define an order vector $\boldsymbol{O} = [o_1 \cdots o_T](o_i = i)$ to express the actual order of $\boldsymbol{x}$.

Definition 2. Let $Z = \{1, 2, \ldots, T\} \subseteq \mathbb{N}^*$, and $G(Z)$ be a group of all bijective transformations on Z. When $|Z| = T$, G is a permutation group of order T, denoted as G_T. For $\forall \sigma_i \in G_T$, we define $\sigma_i = \begin{bmatrix} 1 & 2 & \ldots & T \\ \sigma_i(1) & \sigma_i(2) & \ldots & \sigma_i(T) \end{bmatrix}$, where $[\sigma_i(1) \quad \sigma_i(2) \quad \cdots \quad \sigma_i(T)]$ is a permutation of $[1 \quad 2 \quad \cdots \quad T]$. There are $T!$ permutations, and thus $|G_T| = T!$.

1) Sequence Shuffle: As mentioned in Section IV-B, $\boldsymbol{f}$ is considered as a time series with the same temporal causality as $\boldsymbol{x}$. The sequence shuffle module which disorders $\boldsymbol{f}$ and the corresponding order vector $\boldsymbol{O}$ using the same permutation method can be formalized by Eq. (4):

$$\begin{aligned} \boldsymbol{f_s} &= [\boldsymbol{f}^{\sigma_j(1)} \cdots \boldsymbol{f}^{\sigma_j(T)}]^\top, \\ \boldsymbol{O_s} &= [o_{\sigma_j(1)} \cdots o_{\sigma_j(T)}]^\top, \end{aligned} \tag{4}$$

where $\boldsymbol{f_s}$ and $\boldsymbol{O_s}$ are shuffled feature of $\boldsymbol{f}$ and shuffled order part $\boldsymbol{O}$ under the permutation of $\sigma_j \in G_T$, respectively. For the implementation of the sequence shuffle, we first concatenate $\boldsymbol{O}$ and $\boldsymbol{f}$ at the spatial dimension to form a fusion tensor $\boldsymbol{O_f} \subseteq \mathbb{R}^{T\times(1+D_e)}$. Then, we randomly shuffle the rows of the fusion tensor with the random SEED, which is a variate for each training sample, resulting in the shuffled fusion tensor $\boldsymbol{O_{fs}} \subseteq \mathbb{R}^{T\times(1+D_e)}$. Finally, we obtain $\boldsymbol{O_s} \subseteq \mathbb{N}^T$ by taking $\boldsymbol{O_{fs}}[:, 1]$, and $\boldsymbol{f_s} \subseteq \mathbb{R}^{T\times D_e}$ is obtained by taking $O_{fs}[:, 2 : D_e]$.

The sequence shuffle module enhances the generalization capability of the downstream LRT, as the optimization target of LRT is related to $\boldsymbol{O_s}$ taking the shuffled feature $\boldsymbol{f_s}$ as input. Without the sequence shuffle, the optimization target of LRT would always be related to $\boldsymbol{O}$ for the feature $\boldsymbol{f}$ of every training sample, leading to a reduction in the generalization of LRT.

2) Low-rank Transformer: On the one hand, the training samples for each legitimate user are collected within a limited time. On the other hand, the traditional Transformer involves many parameters that necessitate training on extensive datasets. Therefore, inspired by [38], we design a low-rank Transformer to regress the ranking vector $\boldsymbol{R} \subseteq \mathbb{R}^T$ using the disordered feature $\boldsymbol{f_s}$, as illustrated in Fig. 4. LRT is a typical Transformer encoder architecture, mainly comprising a low-rank multi-head attention module (LRMHA) and a low-rank feed-forward module (LRFF). The details of LRMHA, LRFF, and low-rank encoder-decoder (LED) are presented in Fig 5. In both LRMHA and LRFF, we use LED as a substitute for matrix multiplication in the traditional Transformer to reduce the parameters of the Transformer encoder. In the original Transformer, matrix multiplication maps input data $(T \times m)$ to output data $(T \times n)$ using a parameter matrix $\boldsymbol{W} \subseteq \mathbb{R}^{m\times n}$. In contrast, as illustrated in Fig. 5(c), LED uses a linear encoder

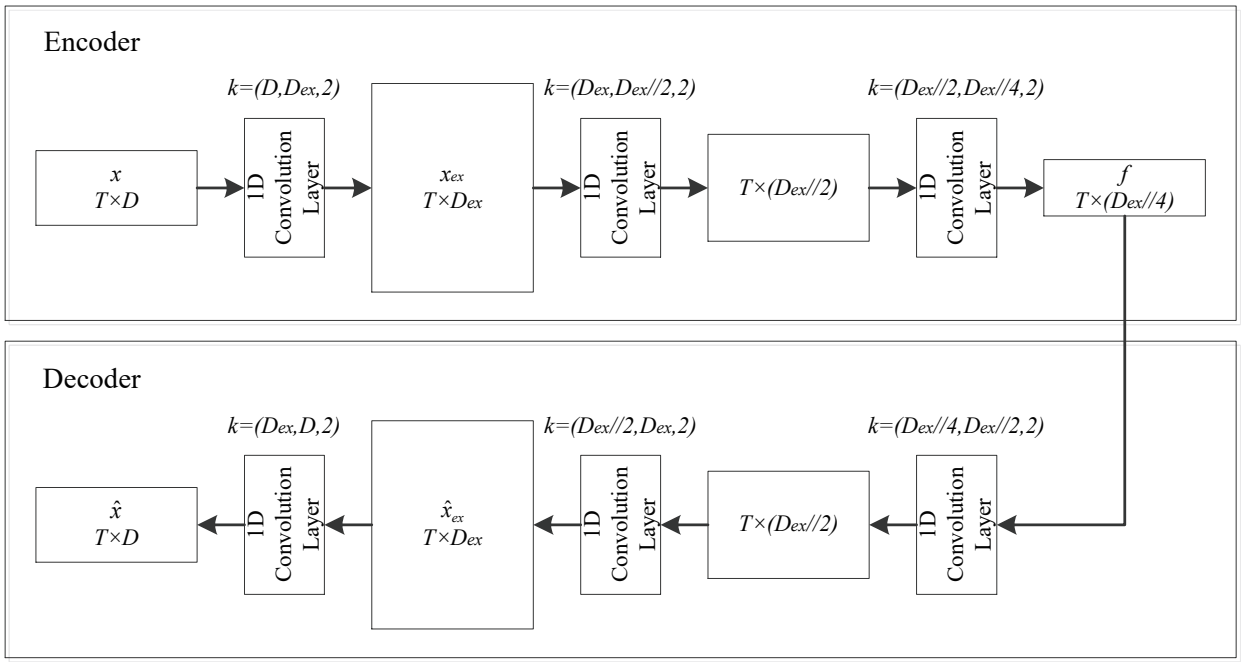


Fig. 3. Architecture of the autoencoder for encoding $\boldsymbol{x}$ and reconstructing $\hat{\boldsymbol{x}}$. The upper portion is the encoder part, which consists of three 1D-convolution layers to compress $\boldsymbol{x}$ into a feature sequence $\boldsymbol{f}$ used by the SLRT module. The lower portion is the decoder part, which comprises three 1D-convolution layers symmetrically arranged with the encoder to reconstruct $\hat{\boldsymbol{x}}$ by $\boldsymbol{f}$. For the kernel $k = (c_{in}, c_{out}, 2)$, c_{in} and c_{out} represent spatial dimensions before and after the 1D-convolution layer, respectively, and 2 indicates the kernel size.

$\boldsymbol{E} \subseteq \mathbb{R}^{m\times r}$ and a linear decoder $\boldsymbol{D} \subseteq \mathbb{R}^{r\times n}$ to approximate $\boldsymbol{W}$, as expressed in Eq. (5):

$$\boldsymbol{W} \approx \boldsymbol{E} \times \boldsymbol{D}, \tag{5}$$

The parameters and floating-point operations (FLOPs) of $\boldsymbol{E}$ and $\boldsymbol{D}$ are $rm + rn = r(m + n)$, while $\boldsymbol{W}$ requires mn parameters and FLOPs. By choosing a considerably smaller value for r than m or n, the parameters of $\boldsymbol{E}$ and $\boldsymbol{D}$ significantly decrease in comparison to $\boldsymbol{W}$.

In LRT, the input feature $\boldsymbol{f_s} = [\boldsymbol{f}_{\sigma_j(1)} \cdots \boldsymbol{f}_{\sigma_j(T)}]$ is first added with positional encode $\boldsymbol{P} = [\boldsymbol{p}_1 \cdots \boldsymbol{p}_T]$ to obtain $\boldsymbol{f_{sp}} = [\boldsymbol{f}_{\sigma_j(1)} + \boldsymbol{p}_1 \cdots \boldsymbol{f}_{\sigma_j(T)} + \boldsymbol{p}_T]$. Then, as shown in Fig. 5(a), LRMHA uses $\boldsymbol{f_{sp}}$ to calculate multi-head attention. It maps $\boldsymbol{f_{sp}}$ into three tensors, i.e., $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$, by three LED units, and each tensor is divided into g heads, i.e., $\boldsymbol{Q} = \{\boldsymbol{Q_i}\}_{i=1}^{g}$, $\boldsymbol{K} = \{\boldsymbol{K_i}\}_{i=1}^{g}$, and $\boldsymbol{V} = \{\boldsymbol{V_i}\}_{i=1}^{g}$. For the ith head, the scaled dot-product attention unit in LRMHA calculates attention by Eq. (6):

$$Attenion(\boldsymbol{Q_i}, \boldsymbol{K_i}, \boldsymbol{V_i}) = Softmax(\frac{\boldsymbol{Q_i}\boldsymbol{K_i^T}}{\sqrt{\boldsymbol{d_q}}})\boldsymbol{V_i}, \tag{6}$$

where d_q indicates the dimension of query. LRMHA further combines the attention from all the g heads to obtain the final attention $\boldsymbol{A} \subseteq \mathbb{R}^{T\times D_e}$ by Eq. (7):

$$\boldsymbol{A} = LED(Dr(Concat(\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_g}))), \tag{7a}$$

$$\boldsymbol{h_i} = Attention(\boldsymbol{Q_i}, \boldsymbol{K_i}, \boldsymbol{V_i}), \tag{7b}$$

where $\boldsymbol{h_i}$ represents the attention for the ith head. After LRMHA, LRT applies dropout to $\boldsymbol{A}$, and the result is added to $\boldsymbol{f_{sp}}$. Then, LRT uses layer normalization (LN), and the normalized feature is further processed by LRFF. As depicted in Fig. 5(b), LRFF comprises two LED units with the ReLU activation function and dropout. The output of LRFF is calculated by Eq. (8):

$$\begin{aligned}\boldsymbol{Output} =& LED_{output}(Dr \\ & (ReLU(LED_{input}(LN(\boldsymbol{A} + \boldsymbol{f_{sp}}))))),\end{aligned} \tag{8}$$

Finally, LRT obtains the ranking vector $\boldsymbol{R}$ by Eq. (9):

$$\begin{aligned}\boldsymbol{R} = \phi(Linear(LN \\ (LN(Dr(\boldsymbol{A}) + \boldsymbol{f_{sp}}) + Dr(\boldsymbol{Output})))),\end{aligned} \tag{9}$$

where ϕ denotes the non-linear activation function $Sigmoid(\cdot)$, and Dr represents a dropout operation. Note that the optimization objective of LRT relies on the information provided by the order part $\boldsymbol{O_s}$ of $\boldsymbol{f_s}$, and the following part elaborates on the design of the training label for SLRT and the training policy.

D. Training Autoencoder and LRT

As mentioned in Section IV-B, the optimization objective of the autoencoder is to reconstruct the input time series sample, and thus, the loss function is formalized as Eq. (10):

$$L_{rec} = Distance(\boldsymbol{x}, \hat{\boldsymbol{x}}), \tag{10}$$

where $\hat{\boldsymbol{x}}$ represents the reconstruction result of a training sample $\boldsymbol{x}$, and $Distance(\cdot, \cdot)$ denotes the measurement (using L_2 distance) of the distance between the two samples. To optimize SLRT, we first design the label by mapping elements of the order part $\boldsymbol{O_s}$ to the interval (0,1] by Eq. (11):

$$\boldsymbol{O_s}/T = [o_{\sigma_j(1)}/T \cdots, o_{\sigma_j(T)}/T], \tag{11}$$

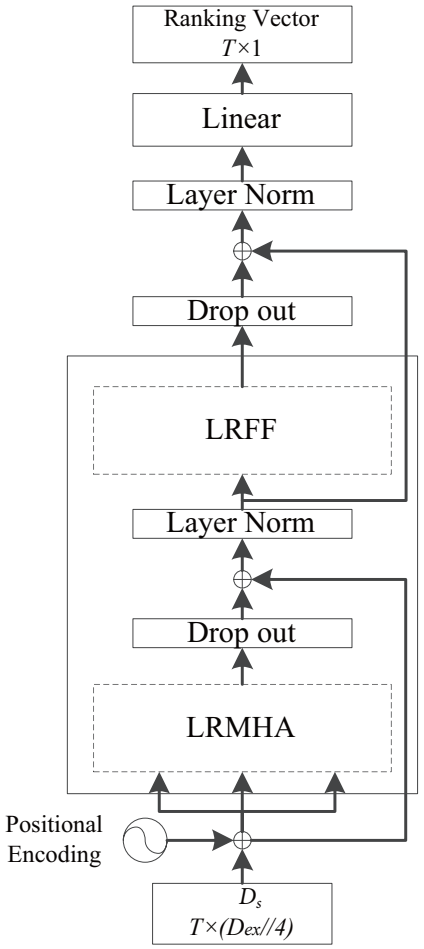


Fig. 4. Architecture of the LRT module. It mainly consists of LRMHA and LRFF, utilized for regressing the ranking vector using the disordered feature f_s of f. In LRT, LRMHA and LRFF correspond to MHA and FF in the traditional Transformer encoder, respectively.

where T denotes the length of the training time series sample. Before the mapping expressed by Eq. (11), each element of $\boldsymbol{O_s}$ is a natural number, while $\boldsymbol{R}$, the output of SLRT, falls within the interval (0,1) due to the sigmoid function of LRT. An alternative solution to circumvent the label mapping is to eliminate the last sigmoid function in LRT and use $\boldsymbol{O_s}$ as the label. However, this alternative approach results in a significantly larger loss compared to L_{rec} and increases the complexity for the converge of LRT. Given the designed label $\boldsymbol{O_s}/T$ and the ranking vector $\boldsymbol{R}$ of SLRT, we formalize the loss function L_{dis} by Eq. (12):

$$L_{dis} = ||\boldsymbol{R} - \boldsymbol{O_s}/T||_2^2, \tag{12}$$

However, the loss calculated by Eq. (12) is hard to reveal and recover the actual order of the disordered data part $\boldsymbol{f_s}$ within $\boldsymbol{f}$. As shown in Table II, we observe that the loss of the model is low, but the ranking vector does not accurately reveal the actual order of the data part.

To alleviate this problem, we design an order consistency loss L_{oc} formalized as Eq. (13):

TABLE II
PERFORMANCE OF SLRT SOLELY OPTIMIZED BY L_{dis}. $\boldsymbol{O_s}$ REPRESENTS THE ORDER PART OF $\boldsymbol{f_s}$, $\boldsymbol{O_s/T}$ AND $\boldsymbol{R}$ INDICATE THE LABELS AND RANKING VECTOR OF SLRT. THE PREDICTED ORDER IS OBTAINED BY MATCHING THE RANKING VECTOR WITH AN ORDER NUMBER (1-4).

$\boldsymbol{O_s}$	$\boldsymbol{O_s/T}$	$\boldsymbol{R}$	Predicted Order	Loss
4,2,3,1	1,0.5,0.75,0.25	0.95,0.69,0.68,0.24	4,3,2,1	0.2088

$$\begin{aligned} L_{oc} &= Rank(\boldsymbol{R}, \boldsymbol{O_s}) \\ &= \sum_{i=1}^{T}\sum_{j=1}^{T} max(0, -(r_i - r_j)(O_{s_i} - O_{s_j})), \end{aligned} \tag{13}$$

where $r_i, r_j \in \boldsymbol{R}$, and $\boldsymbol{O_{s_i}}, \boldsymbol{O_{s_j}} \in \boldsymbol{O_s}$. Recalling that we train the autoencoder and SLRT networks collaboratively, we formalize the total loss L_{total} for CALL as Eq. (14):

$$\begin{aligned} L_{total} &= \mathbb{E}_{\boldsymbol{x}\sim\boldsymbol{X}}[\alpha L_{rec} + \beta L_{dis} + \gamma L_{oc}] \\ &= \frac{1}{N}\sum^{N}[\alpha||Dec(En(\boldsymbol{x})) - \boldsymbol{x}||_2^2 \\ &+ \beta||LRT(S(En(\boldsymbol{x}))) - S(\boldsymbol{O}/T)||_2^2 \\ &+ \gamma Rank(LRT(S(En(\boldsymbol{x}))), S(\boldsymbol{O}/T))], \end{aligned} \tag{14}$$

where α, β, γ $(\alpha + \beta + \gamma = 1)$ indicate penalty factors, N is the total number of training time series samples, Dec and En represent the decoder and encoder of the autoencoder, S denotes the sequence shuffle operation in SLRT, L is the length of each sample, and $\boldsymbol{O}$ is the order part of $En(\boldsymbol{x})$. Therefore, the optimization objective for training CALL is to minimize the total loss L_{total}.

E. Authentication

After trained on $\boldsymbol{X}$, which contains a legitimate user's data, the autoencoder and SLRT in CALL captures spatial features and temporal causality of $\boldsymbol{X}$, respectively. In this section, we detail the REC, SR, and DFW algorithms designed for CALL authentication.

1) Reconstruction Error Calculation: The autoencoder reconstructs $\hat{\boldsymbol{x}}_{\boldsymbol{test}}$ from a test sample $\boldsymbol{x}_{\boldsymbol{test}}$ and calculates L_{rec} using Eq. (15):

$$L_{rec} = ||\hat{\boldsymbol{x}}_{\boldsymbol{test}} - \boldsymbol{x}_{\boldsymbol{test}}||_2, \tag{15}$$

L_{rec} is below or equal to Th_{rec} if $\boldsymbol{x}_{\boldsymbol{test}}$ is legitimate, while L_{rec} is large (beyond Th_{rec}) if the test sample is not from the distribution of $\boldsymbol{X}$. The autoencoder consisting of pure 1D-convolution networks processes only spatial dimensions and maintains temporal causality of $\boldsymbol{x}_{\boldsymbol{test}}$ as expressed by Eq. (3). Therefore, the low L_{rec} indicates that spatial dimensions at every time step of the test sample $\boldsymbol{x}_{\boldsymbol{test}}$ are consistent with the real distribution. However, to demonstrate the legitimate test sample, the similarity in spatial dimensions must be improved. Therefore, as a complement, we use the trained SLRT to capture temporal features of $\boldsymbol{x}_{\boldsymbol{test}}$.

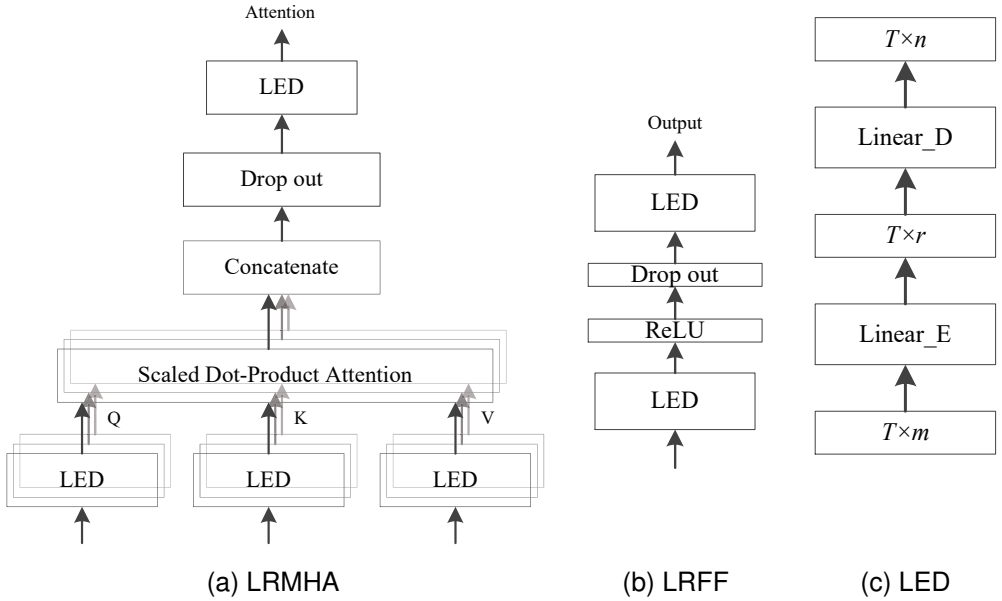


Fig. 5. Architectures of LRMHA, LRFF, and LED for constructing LRT.

2) Sequence Reorder: We design the SR algorithm to reorder $\boldsymbol{x_{s\_test}}$ based on the ranking vector $\boldsymbol{R_{test}}$. The trained SLRT captures the temporal causality of $\boldsymbol{x_{test}} = [\boldsymbol{x_{test}^1} \cdots \boldsymbol{x_{test}^T}]$ by predicting the time order of the shuffled feature $\boldsymbol{f_{test}}$ (obtained from $\boldsymbol{x_{test}}$ after encoder) according to the spatial dimensions of $\boldsymbol{f_{test}}$. Specifically, recall that the sequence shuffle in SLRT disorders the feature and the order part of $\boldsymbol{f_{test}}$ to obtain the unordered feature $\boldsymbol{f_{s\_test}} = [\boldsymbol{f_{test}^{\sigma_k(1)}} \cdots \boldsymbol{f_{test}^{\sigma_k(T)}}]$ and the order part $\boldsymbol{O_{s\_test}} = [o_{\sigma_k(1)} \cdots o_{\sigma_k(T)}]$ of $\boldsymbol{f_{test}}$, respectively. LRT predicts the ranking vector $\boldsymbol{R_{test}}$ by taking $\boldsymbol{f_{s\_test}}$ as input. The permutation information of $\boldsymbol{f_{test}}$ is recorded as SI (related to $\sigma_k \in G_T$), and $\boldsymbol{x_{test}}$ is disordered based on SI to obtain the unordered sample $\boldsymbol{x_{s\_test}} = [\boldsymbol{x_{test}^{\sigma_k(1)}} \cdots \boldsymbol{x_{test}^{\sigma_k(T)}}]$. Particularly, the implementation to obtain the disordered sample $\boldsymbol{x_{s\_test}}$ based on SI is achieved by applying the same random SEED as used for shuffling $\boldsymbol{f_{test}}$ to disorder $\boldsymbol{x_{test}}$.

The goal of SR is to recover the disordered test sample $\boldsymbol{x_{s\_test}}$ according to $\boldsymbol{R_{test}}$ from LRT, where each element r_{test_i} of $\boldsymbol{R_{test}}$ predicts the actual order of $\boldsymbol{x_{test}^{\sigma_k(i)}}$. The recovery steps are as follows: 1) construct an empty vector $\boldsymbol{x_{test\_r}}$; 2) find the minimum element r_{test_j} in $\boldsymbol{R_{test}}$; 3) append $\boldsymbol{x_{test}^{\sigma_k(j)}}$ of $\boldsymbol{x_{s\_test}}$ to $\boldsymbol{x_{test\_r}}$; 4) set r_{test_j} as 1 (recall 1 is the upper bound of elements in $\boldsymbol{R_{test}}$); and 5) loop through steps 2)-4) until the length of $\boldsymbol{x_{test\_r}}$ reaches T. The process of SR is summarized in Algorithm 1, and based on SR, we obtain the reordered sample $\boldsymbol{x_{test\_r}}$. Next, we apply DFW and FFT algorithms to the test sample $\boldsymbol{x_{test}}$ (not shuffled) and $\boldsymbol{x_{test\_r}}$.

3) DFW: We use DFW to calculate the similarity between the frequency spectrum sequences of $\boldsymbol{x_{test_i}^{1:T}}$ and $\boldsymbol{x_{test\_r_i}^{1:T}}$. FFT aims to transform time series data from the time domain to the frequency domain, where the frequency spectrum (amplitude-frequency) can be obtained. Recall that $\boldsymbol{x_{test}} \subseteq \mathbb{R}^{T \times D}$ is multivariate time series data, and for each dimension's sequence $\boldsymbol{x_{test_i}^{1:T}}$, it is processed by FFT as shown in Eq. (16):

Algorithm 1: SR Algorithm

Input: Unordered test sample $\boldsymbol{x_{s\_test}}$, ranking vector $\boldsymbol{R_{test}}$
Output: Recovered test sample $\boldsymbol{x_{test\_r}}$

```
Declare an empty vector x_{test_r}
for iteration 1 to T do
    j ← Index_Of(Min(R_test))
    Append x_test^{σ_k(j)} to x_{test_r}
    r_{test_j} ← 1
end
return x_{test_r}
```

$$F(\omega_k) = \frac{1}{T} \sum_{n=1}^{T} \boldsymbol{x_{test_i}^n} e^{-j\omega_k n} = A_i(\omega_k) \cdot e^{j\theta(\omega_k)}, \quad (16)$$

where ω_k indicates the frequency, and $A_i(\omega_k)$ represents the corresponding amplitude spectrum. Similarly, frequency spectrum of $\boldsymbol{x_{test\_r_i}^{1:T}}$ can also be obtained by Eq. (16), and the amplitude spectrum is denoted as $A_{r_i}(\omega_k)$. The total non-negative frequency K of $\boldsymbol{x_{test_i}^{1:T}}$ and the value of each frequency ω_k (K in total) after FFT are determined by Nyquist frequency principle and the sampling rate. Each frequency ω_k corresponds to a non-negative amplitude $A_i(\omega_k)$ according to Eq. (16), where we can regard the two amplitude vectors $\boldsymbol{A_i} = [A_i(\omega_1) \cdots A_i(\omega_K)]$ and $\boldsymbol{A_{r_i}} = [A_{r_i}(\omega_1) \cdots A_{r_i}(\omega_K)]$ as two sequences. Inspired by DTW, we use the warping algorithm similar to that in DTW (called DFW) to calculate the similarity $L_{sim_i} = DFW(\boldsymbol{A_i}, \boldsymbol{A_{r_i}})$ between these two sequences in each dimension. Finally, we compute the average similarity across all dimensions to obtain the L_{sim} using Eq.

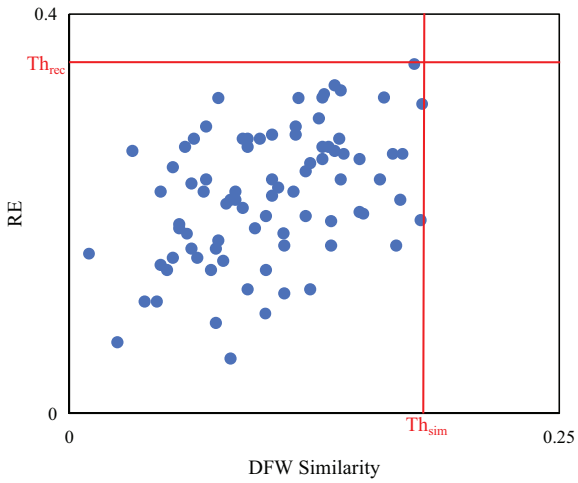


Fig. 6. Visualization of threshold selection. Blue points represent samples taken from the testing fold for determining thresholds.

(17):

$$L_{sim} = \frac{1}{D}\sum_{i=1}^{D} DFW(\boldsymbol{A_i}, \boldsymbol{A_{r_i}}), \tag{17}$$

As mentioned in Section IV-A, after L_{rec} and L_{sim} are obtained by Eq. (15) and Eq. (17), we compare thees values with the predefined thresholds Th_{rec} and Th_{sim}, and then, the corresponding identity of $\boldsymbol{x_{test}}$ can be determined based on these comparison outcomes.

4) Thresholds Selection: The authentication is achieved by comparing L_{rec} and L_{sim} derived from the outputs of CALL, against two thresholds: Th_{rec} and Th_{sim}. Employing N-fold cross-validation for training and testing CALL, we leverage 50% of the testing fold (comprising entirely positive samples) to determine the two thresholds, and the remaining 50% are used to evaluate the performance of CALL. To illustrate the threshold selection process, we present an example in Fig. 6, where the blue points represent samples from the testing fold for determining threshold. We then set the minimum reconstruction error (RE) and DFW similarity values needed to classify all the blue points as positive samples as the thresholds, depicted as the red lines in Fig. 6.

V. Experiments and Results

A. Experimental Settings

In this section, we provide a comprehensive overview of the experimental settings. First, we elaborate on the datasets for the performance evaluation, including our dataset and two public datasets. Then, we detail the preprocessing steps applied on these datasets, comprising denoising, normalization, and sliding window segmentation. Finally, we detail the training and testing policy, and introduce commonly used metrics for the performance evaluation of CALL.

1) Datasets: For **our dataset**, sensor-based behavioral data were collected using the built-in accelerometer, gyroscope, and magnetometer on smartphones by 100 volunteers. Each volunteer was instructed to perform three indoor tasks: document reading, text production, and navigation on a map to locate a destination, across 24 sessions (8 reading, 8 writing, and 8 map navigation) lasting 2 to 6 hours. The sensors' sampling rate was set at $100Hz$. After data collection, we select data from 88 volunteers (excluding 12 with significant noise and partially missing data). For training and testing continuous authentication systems, we extract a total of 100 minutes of data from 24 sessions for each user, basically covering the majority of data for each session. The accelerometer measures the direction of mobile devices and gravitational force in meters per second squared. The gyroscope measures the rate of angle change around the three axes of mobile devices. The magnetometer records magnetic field intensity in the operating environment. The accelerometer data at the ith time step are represented by a vector $\boldsymbol{A_c^i} = [x_a^i \quad y_a^i \quad z_a^i]^\top \subseteq \mathbb{R}^3$, where x_a^i, y_a^i, and z_a^i correspond to the three orthogonal axes. Similarly, the gyroscope and magnetometer data can be denoted as $\boldsymbol{G_r^i} = [x_g^i \quad y_g^i \quad z_g^i]^\top$ and $\boldsymbol{M_a^i} = [x_m^i \quad y_m^i \quad z_m^i]^\top$. We use the fusion vector $\boldsymbol{Fusion^i} = [x_a^i \quad x_g^i \quad x_m^i \quad y_a^i \quad y_g^i \quad y_m^i \quad z_a^i \quad z_g^i \quad z_m^i]^\top \subseteq \mathbb{R}^9$ to represent the sample collected by the three sensors at the ith time step. Finally, for each participant, we obtain the data $\boldsymbol{X} = [\boldsymbol{Fusion^1} \quad \boldsymbol{Fusion^2} \quad \cdots \quad \boldsymbol{Fusion^{600000}}]^\top \subseteq \mathbb{R}^{600000\times 9}$.

UCI_HAR dataset [39] was collected using smartphones worn around the waist by 30 participants aged 19 to 48. Participants were instructed to perform six daily activities: walking, walking upstairs, walking downstairs, sitting, standing, and lying down. The smartphone's sampling rate was set at 50 Hz, and the recorded sensor data included the triaxial linear acceleration and triaxial angular velocity. For this dataset, sensor data from 9 individuals were excluded (user IDs: $2, 4, 9, 10, 12, 13, 18, 20$, and 24), resulting in this dataset comprising information from 21 participants.

WISDM_HARB dataset [40] was collected using smartphones and smartwatches worn by 51 participants. Participants were instructed to conduct 18 specified activities within three minutes each day, consisting of 5 simple and 13 complex activities. In this dataset, data are collected by the accelerometer and gyroscope at a consistent rate of 20 Hz from smartphones and smartwatches. According to the study in [41], 7 participants (user IDs: $1616, 1618, 1637, 1638, 1639, 1640$, and 1642) were identified as abnormal, resulting in this dataset containing information from 44 participants.

2) Data Preprocessing: For our dataset, based on the 88 participants' data, we first employ a low-pass filter for denoising, and then normalize each participant's data X. For the ith spatial dimension $X_i^{1:600000} \subseteq \mathbb{R}^{600000\times 1}$, we apply min-max normalization using the formula: $\frac{X_i^j - Min\{X_i^t\}_{t=1}^{600000}}{Max\{X_i^t\}_{t=1}^{600000} - Min\{X_i^t\}_{t=1}^{600000}}$. Following normalization, we utilize a sliding window with a size of 200 and a sliding step of 200 to segment the normalized X into $\frac{600000}{200} = 3000$ windows, denoted as $X = \{x_1, x_2, \ldots, x_{3000}\}$ ($x_i \subseteq \mathbb{R}^{200\times 9}$). Thus, each participant's data in our dataset are divided into 3000 time series samples. For UCI_HAR and WISDM_HARB, which are benchmark datasets, we set the size of the data window as 2.56 seconds. After normalization identical to that of our dataset, the profile of the three datasets

TABLE III
PROFILE OF UCI_HAR, WISDM_HARB AND OUR DATASETS AFTER PREPROCESSING.

Dataset	User	Length of Data Window	Dimension
UCI_HAR	21	128	6
WISDM_HARB	44	52	6
Ours	88	200	9

we use is presented in Table III.

3) Training Settings: We detail the training and testing policy, employing the five-fold cross-validation method. Assuming there are N users in a dataset, one is regarded as the legitimate user, and the remaining $N-1$ users are considered as imposters. The data from the legitimate user, denoted as X, are used for training, and a five-fold cross-validation strategy is adopted to train and test CALL. Particularly, imposters' data T_{im} equivalent to 20% of X (one fold) are randomly selected from the rest $N-1$ users. These imposter samples T_{im} are then appended to each testing fold to evaluate the performance of CALL. Thus, for each dataset involving N potential legitimate users in our experiments, N CALL systems can be trained and N metric values (e.g., the accuracy) can be obtained. Finally, we calculate the mean of the N values from the N trained systems as the final performance result.

4) Metrics: For our continuous authentication system CALL, we employ three metrics to evaluate the performance in each testing fold appended with T_{im}: accuracy (ACC)= $\frac{TP+TN}{TP+TN+FP+FN}$, F_1_Score= $\frac{2 \cdot TP}{2 \cdot TP+FP+FN}$, and EER. In particular, EER is a comprehensive indicator representing the balance between the security and convenience of using continuous authentication systems, and it is the point where FAR= $\frac{FP}{FP+TN}$ equals to FRR= $\frac{FN}{FN+TP}$.

B. Overall Performance

1) Fundamental Results: In this section, we present fundamental results of CALL on the three datasets. Specifically, we first provide training loss curves on the three datasets to demonstrate convergence. Then, employing SR in Algorithm 1 and FFT, we visualize sequence diagrams and frequency spectrum sequences for the original legitimate data v.s. the reordered legitimate data, and the original imposter's data v.s. the reordered imposter's data, to illustrate the effectiveness of CALL.

Loss curve. We show loss curves for CALL trained on a randomly selected legitimate user from the three datasets in Fig. 7. As shown, all the curves converge to a small value, i.e., 0.0016 on our dataset, 0.01 on UCI_HAR, and 0.09 on WISDM_HARB. Furthermore, the loss curves trained on all the other users exhibit a similar convergence trend to that shown in Fig. 7.

Sequence diagram and frequency spectrum. For each selected legitimate user X trained on the three datasets, we can visualize a sequence diagram of one dimension within a data window from the testing fold and show its FFT and inverse FFT (IFFT) results. Taking our dataset as an example, the visualization is illustrated in Fig. 8(a). Additionally, after CALL is trained on X, we use CALL to process the same data window of X to obtain a ranking vector and use Algorithm 1 to reorder this data window, where we also provide sequence diagram, IFFT, and FFT results. We then compare the results from the original data window of X with the reordered data, as depicked in Fig. 8(b). On the one hand, we observe that the sequence diagram of the reordered data globally exhibits significant difference from the original data window, whereas the ideal outcome would be for the reordered data to follow the same trend as the original data window, as per the loss function in Eq. (14). On the other hand, when focusing on the local periods of the reordered data bounded by dashed boxes, we find that each period follows the same trend as the original data. Furthermore, after FFT, it becomes evident that the frequency spectrum sequences of the reordered and the original data are almost identical in shape, and the DFW distance is small. Particularly, this pattern holds for all the other dimensions of the data window and different data windows on the three datasets. Fig. 8 reveals the effectiveness of DFW distance for CALL, as the reordered data exhibit several periods (dashed boxes) resembling the trend of original data. Thus, the former can be regarded as a periodic signal changed from the latter with high frequency, resulting in the latency of the peaks and valleys with the same amplitude for the two sequences in frequency spectrum after FFT. Therefore, DFW performs well in calculating the two misaligned frequency-spectrum sequences.

We then randomly select an imposter X_{im} from our dataset, which is from the remaining $N-1$ users, as mentioned in Section V-A3. Using the similar methods as described above, we compare the sequence diagrams, IFFT, and FFT results of the original data window from X_{im} and the reordered one processed by CALL, as illustrated in Fig. 9. We observe that the sequence diagrams of the original and the reordered data are similar over a large time range (approximately $0-1230ms$ in Fig. 9(b)). However, the frequency spectrum of the reordered one differs significantly from that of the original data, and the distance between the original and the reordered data is large, thereby validating the effectiveness of SLRT. Particularly, this pattern holds for all the other dimensions as well.

2) Comparison with SOTA Methods: We compare our unsupervised CALL system with three supervised and one unsupervised state-of-the-art (SOTA) sensor-based continuous authentication systems, i.e., supervised LSTMAuth (LSTMAuth-S) [23], GMM-UBM [42] and 1D-ResNet-SE [24], and unsupervised LSTMAuth (LSTMAuth-U) [23] on the three datasets, i.e., UCI_HAR, WISDM_HARB, and our dataset. The average accuracy and EER performance of CALL and the other three systems is listed in Table IV.

We observe that CALL consistently outperforms the other SOTA continuous authentication systems in terms of accuracy and EER across all datasets. Specifically, on UCI_HAR, WISDM_HARB and our dataset, CALL achieves the highest accuracy of 96.43%, 95.24% and 96.92%, and the lowest EERs of 4.28%, 4.76% and 3.86%, respectively. The superior performance of CALL on our dataset can be attributed to its larger size compared to the other two public datasets.

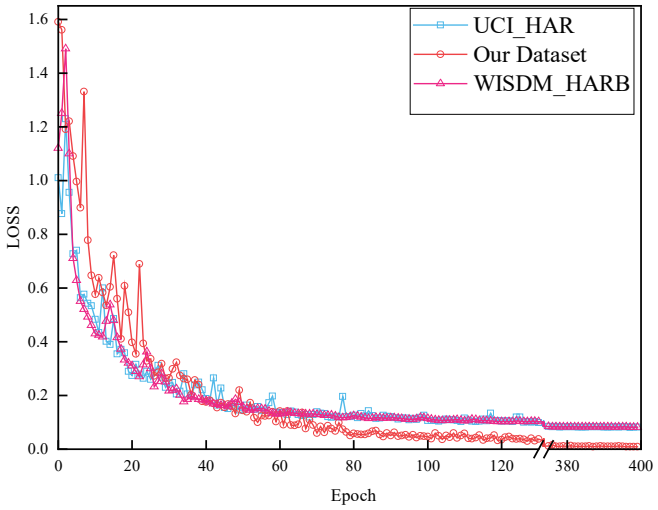


Fig. 7. Loss curves of CALL on UCI_HAR, WISDM_HARB, and our dataset.

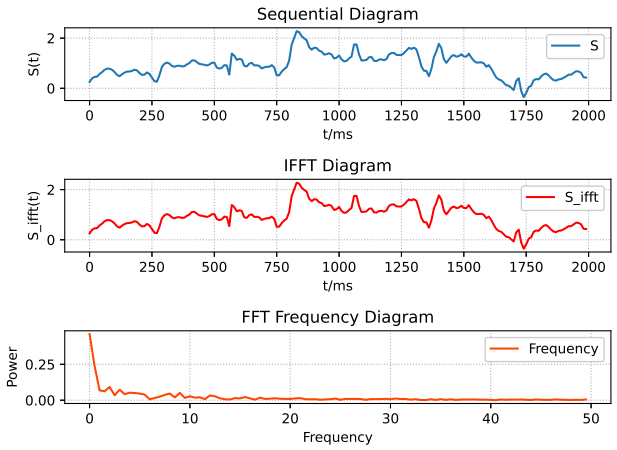
TABLE IV
PERFORMANCE COMPARISON AMONG CALL AND THE THREE SOTA CONTINUOUS AUTHENTICATION SYSTEMS ON UCI_HAR, WISDM_HARB AND OUR DATASET IN TERMS OF ACCURACY (%) AND EER (%).

Dataset	System	ACC	EER
UCI_HAR	GMM-UBM	-	16.50
	LSTMAuth-S	90.10	8.80
	LSTMAuth-U	65.40	20.00
	CALL	96.43	4.28
WISDM_HARB	1D-ResNet-SE	84.77	16.05
	LSTMAuth-S	84.40	9.95
	LSTMAuth-U	75.50	22.60
	CALL	95.24	4.76
Our Dataset	LSTMAuth-S	89.40	13.40
	LSTMAuth-U	76.00	25.00
	CALL	96.92	3.86

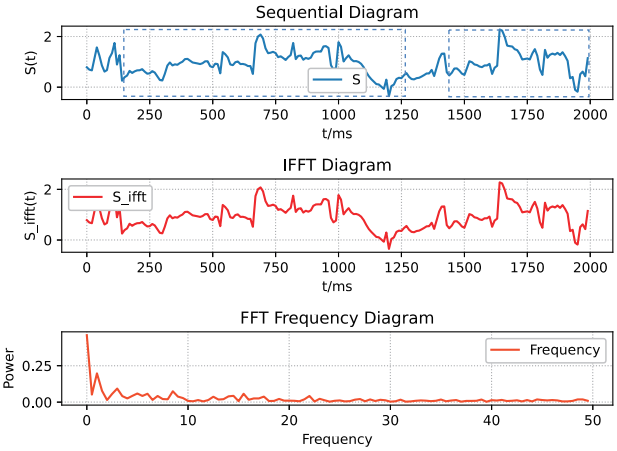
Moreover, for each dataset, CALL is trained solely using data from one user as the legitimate user and is tested utilizing the remaining users' data as unseen imposter's samples, as shown in Section V-A3. In contrast, other supervised systems leverage a substantially larger volume of users' data for training, which demonstrates that CALL achieves superior performance despite being trained with a smaller dataset. In particular, we also observe that the average accuracy of 1D-ResNet-SE (84.77%) slightly surpasses that of LSTMAuth-S (84.40%), although LSTMAUth-S has a greater number of training samples after data augmentation than 1D-ResNet-SE without any data augmentation.

C. Ablation Study

1) Effect of the Shuffle Operation: We remove the shuffle module from SLRT while keeping the other components



(a) Original data of a legitimate user.

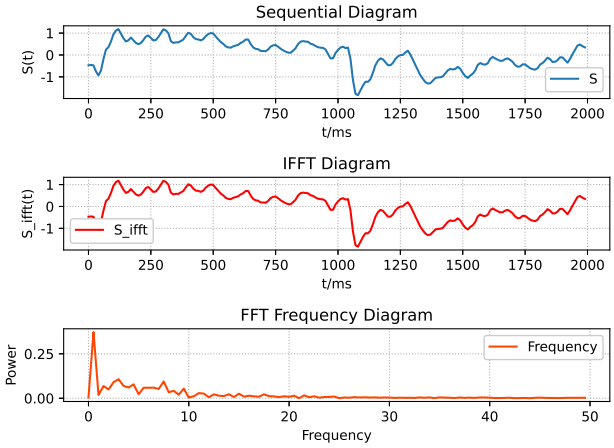


(b) Reordered data of a legitimate user.

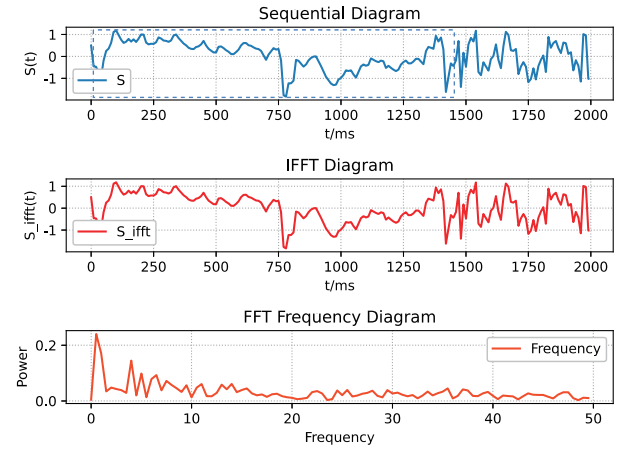
Fig. 8. Comparison of an original data sequence and reordered data sequence for one dimension within a time window on our dataset for a legitimate user.

unchanged. After training for 400 epochs, we compare the average performance of CALL without (w/o) the shuffle module to CALL, and the results are listed in Table V. We observe that CALL without shuffle module can quickly converge during training on each dataset. However, its performance on the three datasets is significantly worse than CALL, with lower accuracy (58.57%, 63.10%, and 66.58%) and higher EERs (38.57%, 44.83%, and 42.52%). Further experiments reveal that CALL without the shuffle model tends to learn fixed outputs since labels for all the training samples are the same (a vector with elements from $1/T$ to 1). This emphasizes the importance of the shuffle operation in CALL to prevent the model from learning meaningless patterns.

2) Effect of OC Loss: We remove the OC loss mentioned in Eq. (13) while keeping the other loss terms unchanged. We compare the average performance of CALL without OC loss to CALL, and the results are presented in Table VI. We observe that CALL without OC loss can converge quickly, but its accuracy (90.00%, 89.29%, and 91.42%) and EERs (9.37%, 13.51%, and 10.91%) are not as good as CALL on the three datasets. By observing the outputs of CALL without OC loss, we further observation finds that CALL without OC loss learns



(a) Original data of an imposter.



(b) Reordered data of an imposter.

Fig. 9. Comparison of an original data sequence and reordered data sequence for one dimension within a time window on our dataset for an imposter.

TABLE V
PERFORMANCE COMPARISON BETWEEN CALL WITHOUT (W/O) SHUFFLE MODULE AND CALL ON UCI_HAR, WISDM_HARB AND OUR DATASET IN TERMS OF AVERAGE ACCURACY (%), F_1_Score (%), AND EER (%).

Dataset	Version	ACC	F_1_Score	EER
UCI_HAR	w/o shuffle module	58.57	57.35	38.57
	CALL	96.43	96.40	4.28
WISDM_HARB	w/o shuffle module	63.10	60.76	44.83
	CALL	95.24	95.24	4.76
Our Dataset	w/o shuffle module	66.58	65.34	42.52
	CALL	96.92	96.88	3.86

to predict the overall order of the time series sample but fails to accurately predict the order of data in specific local areas, similar to the example in Table II. This experiment indicates that OC loss is crucial for helping CALL learn the precise overall and local order of time series data.

3) Effect of the Low-Rank Transformer: To thoroughly investigate the effect of the low-rank Transformer on CALL, we conduct three ablation experiments on three datasets for

TABLE VI
PERFORMANCE COMPARISON BETWEEN CALL WITHOUT (W/O) OC LOSS AND CALL ON UCI_HAR, WISDM_HARB, AND OUR DATASET IN TERMS OF ACCURACY (%), F_1_Score (%) AND EER (%).

Dataset	Version	ACC	F_1_Score	EER
UCI_HAR	w/o OC loss	90.00	89.86	9.37
	CALL	96.43	96.40	4.28
WISDM_HARB	w/o OC loss	89.29	89.16	13.51
	CALL	95.24	95.24	4.76
Our Dataset	w/o OC loss	91.42	91.31	10.91
	CALL	96.92	96.88	3.86

comparison: 1) We remove the SLRT module from CALL, resulting in CALL with (w/) the pure-CNN autoencoder that exclusively captures spatial features expressed by Eq. (3); 2) We replace the low-rank Transformer in SLRT with the traditional Transformer [35], resulting in CALL with traditional Transformer; 3) we replace the low-rank Transformer in SLRT with RNN, resulting in CALL with RNN. We compare the average performance among CALL with pure-CNN autoencoder, with the traditional Transformer, and with RNN, and the results are presented in Table VII. We observe that CALL generally exhibits the best performance on UCI_HAR and WISDM_HARB datasets, outperforming our dataset, where CALL with traditional Transformer shows the best performance. Specifically, CALL with pure-CNN autoencoder significantly degrades, with low accuracy (70.77%, 74.17%, and 78.17%) and high EERs (29.23%, 25.83%, and 20.67%) on three datasets. CALL with RNN cannot work well on all the datasets since it fails to capture accurate temporal relations of long-term multivariate time series data. CALL (with the low-rank-Transformer) achieves the best performance on UCI_HAR and WISDM_HARB, which are small datasets. In contrast, CALL with traditional Transformer obtains the highest accuracy of 97.17% and the lowest EER of 3.27% on our dataset with 2400 training samples for each user. However, the low-rank Transformer in CALL, which has fewer parameters, is proved to perform well on datasets with a small volume of training samples. In comparison, the traditional Transformer, which has more parameters, performs better on the relatively large dataset. In real scenarios, the low-rank Transformer can be applied in the SLRT module with limited training data, while the traditional Transformer can be utilized with the large volume of training data.

4) Effect of the Dataset Size: We divide the training sets into 25%, 50%, 75%, and 100% subsets of the training data used in Section V-A3 for all datasets. For each dataset, The average performance of CALL on each subset is presented in Table VIII. We observe that CALL can achieve high performance on all three datasets, even when the data size is only 50% in 89.29%, 91.67%, and 32.33% accuracy, respectively. Furthermore, it reaches 92.86%, 94.05%, and 95.58% accuracy, and 7.69%, 7.69%, and 5.08% EERs on three datasets with 75% training samples, which is closely matches the performance with 100% training samples. Thus,

TABLE VII
PERFORMANCE COMPARISON AMONG CALL WITH (W/) PURE-CNN AUTOENCODE, WITH TRADITIONAL-TRANSFORMER, WITH RNN, AND CALL ON UCI_HAR, WISDM_HARB AND OUR DATASET IN TERMS OF AVERAGE ACCURACY (%), F_1_Score (%) AND EER (%).

Dataset	Version	ACC	F_1_Score	EER
UCI_HAR	w/ pure-CNN autoencoder	70.77	71.64	29.23
	w/ traditional Transformer	93.57	93.53	6.06
	w/ RNN	76.43	75.56	24.29
	CALL	96.43	96.40	4.28
WISDM_HARB	w/ pure-CNN autoencoder	74.17	74.80	25.83
	w/ traditional Transformer	92.86	92.68	7.69
	w/ RNN	72.62	71.60	28.57
	CALL	95.24	95.24	4.76
Our Dataset	w/ pure-CNN autoencoder	78.17	77.91	20.67
	w/ traditional Transformer	97.17	97.15	3.27
	w/ RNN	80.25	79.93	19.83
	CALL	96.92	96.88	3.86

TABLE VIII
CALL PERFORMANCE COMPARISON AMONG DIFFERENT DATASET SIZES ON UCI_HAR, WISDM_HARB, AND OUR DATASET IN TERMS OF AVERAGE ACCURACY(%), F_1_Score (%) AND EER (%).

Dataset	Percentage (%)	ACC	F_1_Score	EER
UCI_HAR	25	80.00	79.41	21.43
	50	89.29	89.21	11.11
	75	92.86	92.75	7.69
	100	96.43	96.40	4.82
WISDM_HARB	25	73.81	73.17	23.81
	50	91.67	91.36	10.53
	75	94.05	93.98	7.69
	100	95.24	95.24	4.76
Our Dataset	25	85.58	85.40	16.50
	50	92.33	92.23	8.50
	75	95.58	95.54	5.08
	100	96.92	96.88	3.86

despite shrinking the training dataset size to 50%, the random shuffle operation in the feature space still can effectively disorder the features, treating them as different sequences in different training epochs, resulting in theoretically infinite training samples, especially when the data volume is limited. This experiment validates that data augmentation methods are not necessary for CALL.

5) Effect of the sensor numbers: We study the impact of varying numbers of sensors in each dataset on CALL. For each dataset, considering a specified number of sensors (ranging from one to all sensors), there are several possible sensor combinations. We calculate the average performance across all these sensor combinations for each given sensor number. The results are summarized in Table IX. The table illustrates that CALL is sensitive to the number of sensors, with a decrease in performance as the number of sensors is reduced. For instance, with only one sensor, CALL receives 55.00%, 63.10%, and 60.17% accuracy, and 44.29%, 38.10%, and 39.33% EERs, on the three datasets, respectively. Therefore, we conclude that the number of sensors significantly influences CALL, due to the average mutual information. Assuming n sensor sets $\Omega = \{S_1, \ldots, S_n\}$ and one ranking set $T = \{t_1, \ldots, t_m\}$, the information entropy of T for a user under the measurement of one sensor can be formalized as Eq. (18):

$$H(T|S_i) = \sum_{t\sim T, s_i \sim S_i} p(t, s_i) log(\frac{1}{p(t|s_i)}), \tag{18}$$

where $S_i \in \Omega$. We define $I(T; S_i)$ as the average mutual information between T and S_i. Eq. (18) can be modified as Eq. (19):

$$\begin{aligned} H(T|S_i) &= H(T) - I(T; S_i) \\ &= H(T) - \sum_{t\sim T, s_i \sim S_i} p(t, s_i) log(\frac{p(t, s_i)}{p(t)p(s_i)}), \end{aligned} \tag{19}$$

where $H(T)$ denotes the information entropy of T. We know $I(T; S_i) \geq 0$, and the equal sign holds if and only if T and S_i are independent. As a result, the uncertainty (information entropy) is reduced when introducing the measurement of S_i, which allows CALL to predict the order of time series based on the sensor values. However, in real scenarios, the uncertainty of T remains large based solely on one sensor. For example, when a mobile device is used for a certain period, the same value measured by the accelerometer is likely to appear many times during this period uncertainly, which degrades CALL's ability to predict the accurate occurrence time only based on the measured values. Thus, with the same methods by Eq. (19), we use more sensors at each time step to reduce the uncertainty of T, as Eq. (20):

$$\begin{aligned} &H(T|S_1, S_2, \ldots, S_n) = H(T|S_1, S_2, \ldots, S_{n-1}) \\ &- I(T; S_n|S_1, S_2, \ldots, S_{n-1}) \\ &= H(T|S_1, S_2, \ldots, S_{n-2}) - I(T; S_{n-1}|S_1, S_2, \ldots, S_{n-2}) \\ &- I(T; S_n|S_1, S_2, \ldots, S_{n-1}) \\ &\ldots \\ &= H(T|S_1) - I(T; S_2|S_1) - \cdots - I(T; S_n|S_1, S_2, \ldots, S_{n-1}) \\ &= H(T) - I(T; S_1) - \cdots - I(T; S_n|S_1, S_2, \ldots, S_{n-1}), \end{aligned} \tag{20}$$

Due to the nonnegativity of $I(\cdot;\cdot)$, we can observe that $H(T) \geq H(T|S_1) \geq \cdots \geq H(T|S_1, S_2, \ldots, S_n)$. According to Table IX, when the sensor number increases to 2 or more, the uncertainty of the ranking for time series data is significantly reduced, enabling CALL to precisely predict the order based on the measurements.

D. Complexity Analysis

In this section, we analyze the complexity of CALL by comparing it to CALL with the traditional Transformer in terms of floating-point operations per second (FLOPS), parameters, and real-world time cost using the "thop.profile" and "time" Python library on the three datasets, and the results are presented in Table X. We observe that CALL has significantly fewer FLOPS, i.e., $1749.60K$, $586.80K$, and $3661.20K$ on UCI_HAR, WISDM_HARB, and our dataset, and fewer parameters, i.e., $8.95K$, $3.17K$, and $18.57K$ on the three datasets, respectively, which indicate that CALL requires less memory. In addition, for the real-world time cost, we

TABLE IX
PERFORMANCE OF CALL WITH DIFFERENT NUMBER OF SENSORS ON UCI_HAR, WISDM_HARB AND OUR DATASET IN TERMS OF ACCURACY (%), F_1_Score (%) AND EER (%).

Dataset	Sensor Number	ACC	F_1_Score	EER
UCI_HAR	1	55.00	55.32	44.29
	2	96.43	96.40	4.28
WISDM_HARB	1	63.10	63.53	38.10
	2	95.24	95.24	4.76
Our Dataset	1	60.17	59.97	39.33
	2	90.00	89.76	10.33
	3	96.92	96.88	3.86

TABLE X
COMPLEXITY COMPARISON BETWEEN CALL WITH (W/) TRADITIONAL TRANSFORMER AND CALL ON UCI_HAR, WISDM_HARB, AND OUR DATASET IN TERMS OF FLOPS (K), PARAMETERS (K), AVERAGE TIME COST BY MODEL (MS) AND AVERAGE TIME COST BY FFT (MS).

Dataset	Version	FLOPS	Parameter	Model Time	FFT Time
UCI_HAR	w/ traditional Transformer	2171.88	11.10	92.27	1.25
	CALL	1749.60	8.95	60.30	1.26
WISDM_HARB	w/ traditional Transformer	752.06	4.25	49.28	1.41
	CALL	586.80	3.17	37.47	1.43
Our Dataset	w/ traditional Transformer	4256.02	21.81	157.30	2.04
	CALL	3661.20	18.57	107.83	1.92

perform the calculation on CPU, testing 10 samples for each dataset, and repeating the testing 10 times to obtain the average time. We find that CALL takes a lower average time than CALL with the traditional transformer on all datasets, i.e., $60.30ms$, $37.47ms$, and $107.83ms$, which indicates CALL is more suitable for deployment on mobile devices than the CALL with traditional Transformer.

VI. CONCLUSION

This work proposes a lightweight unsupervised sensor-based continuous authentication system using a pure-CNN autoencoder and a designed low-rank Transformer with a learning-to-rank policy, namely CALL, for achieving end-to-end authentication. CALL uses the autoencoder and low-rank Transformer to capture spatial and temporal features from time series data, achieving the best authentication performance through the reconstruction error calculation, SR and DFW algorithms in terms of 96.43%, 95.24%, and 96.92% accuracy, and 4.28%, 4.76%, and 3.86% EERs on UCI_HAR, WISDM_HARB, and our dataset, respectively, comparing to SOTA systems. The experiments further demonstrate that the low-rank Transformer enables CALL to maintain high performance with fewer parameters, reduced FLOPS, and decreased average time cost, making it suitable for mobile devices. Finally, a mathematical proof is provided to explain why CALL exhibits sensitivity to the number of sensors.

In comparison with recent supervised systems trained on extensive datasets, there is still room for improvement in the accuracy and EERs of CALL. This is because we currently employ only one layer of low-rank Transformer and a feed-forward network within the Transformer with 72 dimensions. Furthermore, CALL needs to enhance both convenience and security to offer real-time mobile device authentication. In addition, recent studies suggest that incorporating statistical features can enhance the system's robustness [43]. To address these concerns, our future work will focus on enhancing each module's backbone to reduce inference latency, optimizing hyperparameters, improving the performance of the unsupervised authentication system, and exploring the integration of manual features to promote robustness. We aim to keep the performance of unsupervised models competitive with recent large models. Moreover, our efforts will extend beyond preventing forgery attacks by aiming to authenticate users at each time step, thus achieving higher real-time properties.

REFERENCES

[1] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *4th USENIX Workshop on Offensive Technologies (WOOT 10)*, 2010.

[2] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget, "Design and evaluation of a shoulder-surfing resistant graphical password scheme," in *Proceedings of the working conference on Advanced visual interfaces*, 2006, pp. 177–184.

[3] M. Zhou, Q. Wang, J. Yang, Q. Li, F. Xiao, Z. Wang, and X. Chen, "Patternlistener: Cracking android pattern lock using acoustic signals," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1775–1787.

[4] D. Cheng, L. Zhang, C. Bu, H. Wu, and A. Song, "Learning hierarchical time series data augmentation invariances via contrastive supervision for human activity recognition," *Knowledge-Based Systems*, vol. 276, p. 110789, 2023.

[5] D. Cheng, L. Zhang, C. Bu, X. Wang, H. Wu, and A. Song, "Protohar: Prototype guided personalized federated learning for human activity recognition," *IEEE Journal of Biomedical and Health Informatics*, 2023.

[6] S. Xu, L. Zhang, Y. Tang, C. Han, H. Wu, and A. Song, "Channel attention for sensor-based activity recognition: Embedding features into all frequencies in dct domain," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[7] C. Bu, L. Zhang, H. Cui, G. Yang, and H. Wu, "Dynamic inference via localizing semantic intervals in sensor data for budget-tunable activity recognition," *IEEE Transactions on Industrial Informatics*, 2023.

[8] D. M. Shila and K. Srivastava, "Castra: Seamless and unobtrusive authentication of users to diverse mobile services," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4042–4057, 2018.

[9] I. Lamiche, G. Bin, Y. Jing, Z. Yu, and A. Hadid, "A continuous smartphone authentication method based on gait patterns and keystroke dynamics," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 4417–4430, 2019.

[10] S. Mondal and P. Bours, "A continuous combination of security & forensics for mobile devices," *Journal of information security and applications*, vol. 40, pp. 63–77, 2018.

[11] R. Kumar, V. V. Phoha, and A. Serwadda, "Continuous authentication of smartphone users by fusing typing, swiping, and phone movement patterns," in *2016 IEEE 8th international conference on biometrics theory, applications and systems (BTAS)*. IEEE, 2016, pp. 1–8.

[12] T.-Y. Chang, C.-J. Tsai, J.-Y. Yeh, C.-C. Peng, and P.-H. Chen, "New soft biometrics for limited resource in keystroke dynamics authentication," *Multimedia Tools and Applications*, vol. 79, pp. 23 295–23 324, 2020.

[13] J. Kim and P. Kang, "Freely typed keystroke dynamics-based user authentication for mobile devices based on heterogeneous features," *Pattern Recognition*, vol. 108, p. 107556, 2020.

[14] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi, "Tips: Context-aware implicit user identification using touch screen in uncontrolled environments," in *Proceedings of the 15th workshop on mobile computing systems and applications*, 2014, pp. 1–6.

[15] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE transactions on information forensics and security*, vol. 8, no. 1, pp. 136–148, 2012.

[16] S. Keykhaie and S. Pierre, "Lightweight and secure face-based active authentication for mobile users," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1551–1565, 2023.

[17] Y. Yang, X. Huang, J. Li, and J. S. Sun, "Bubblemap: Privilege mapping for behavior-based implicit authentication systems," *IEEE Transactions on Mobile Computing*, vol. 22, no. 8, pp. 4548–4562, 2023.

[18] W. H. Khoh, Y. H. Pang, A. B. J. Teoh, and S. Y. Ooi, "In-air hand gesture signature using transfer learning and its forgery attack," *Applied Soft Computing*, vol. 113, p. 108033, 2021.

[19] G. Dahia, L. Jesus, and M. Pamplona Segundo, "Continuous authentication using biometrics: An advanced review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 4, p. e1365, 2020.

[20] Y. Li, L. Liu, S. Deng, H. Qin, M. A. El-Yacoubi, and G. Zhou, "Memory-augmented autoencoder based continuous authentication on smartphones with conditional transformer gans," *IEEE Transactions on Mobile Computing*, 2023.

[21] Y. Li, L. Liu, H. Qin, S. Deng, M. A. El-Yacoubi, and G. Zhou, "Adaptive deep feature fusion for continuous authentication with data augmentation," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5690–5705, 2023.

[22] Y. Li, P. Tao, S. Deng, and G. Zhou, "Deffusion: Cnn-based continuous authentication using deep feature fusion," *ACM Transactions on Sensor Networks (TOSN)*, vol. 18, no. 2, pp. 1–20, 2021.

[23] G. Giorgi, A. Saracino, and F. Martinelli, "Using recurrent neural networks for continuous authentication through gait analysis," *Pattern Recognition Letters*, vol. 147, pp. 157–163, 2021.

[24] S. Mekruksavanich and A. Jitpattanakul, "Deep residual network for smartwatch-based user identification through complex hand movements," *Sensors*, vol. 22, no. 8, p. 3094, 2022.

[25] Y. Cao, F. Li, Q. Zhang, S. Yang, and Y. Wang, "Towards nonintrusive and secure mobile two-factor authentication on wearables," *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 3046–3061, 2023.

[26] S. Ahmad, S. Mishra, F. J. Zareen, and S. Jabin, "Sensor-enabled biometric signature-based authentication method for smartphone users," *International Journal of Biometrics*, vol. 15, no. 2, pp. 212–232, 2023.

[27] F. H. Al-Naji and R. Zagrouba, "A survey on continuous authentication methods in internet of things environment," *Computer Communications*, vol. 163, pp. 109–133, 2020.

[28] M. Abuhamad, A. Abusnaina, D. Nyang, and D. Mohaisen, "Sensor-based continuous authentication of smartphones' users using behavioral biometrics: A contemporary survey," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 65–84, 2020.

[29] R. Spolaor, Q. Li, M. Monaro, M. Conti, L. Gamberini, and G. Sartori, "Biometric authentication methods on smartphones: A survey." *PsychNology Journal*, vol. 14, no. 2, 2016.

[30] I. Stylios, S. Kokolakis, O. Thanou, and S. Chatzis, "Behavioral biometrics & continuous user authentication on mobile devices: A survey," *Information Fusion*, vol. 66, pp. 76–99, 2021.

[31] M. Hu, K. Zhang, R. You, and B. Tu, "Authconformer: Sensor-based continuous authentication of smartphone users using a convolutional transformer," *Computers & Security*, vol. 127, p. 103122, 2023.

[32] M. Abuhamad, T. Abuhmed, D. Mohaisen, and D. Nyang, "Autosen: Deep-learning-based implicit continuous authentication using smartphone sensors," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5008–5020, 2020.

[33] P. Delgado-Santos, R. Tolosana, R. Guest, R. Vera-Rodriguez, F. Deravi, and A. Morales, "Gaitprivacyon: Privacy-preserving mobile gait biometrics using unsupervised learning," *Pattern Recognition Letters*, vol. 161, pp. 30–37, 2022.

[34] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[37] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *arXiv preprint arXiv:1901.07291*, 2019.

[38] G. I. Winata, S. Cahyawijaya, Z. Lin, Z. Liu, and P. Fung, "Lightweight and efficient end-to-end speech recognition using low-rank transformer," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6144–6148.

[39] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz *et al.*, "A public domain dataset for human activity recognition using smartphones." in *Esann*, vol. 3, 2013, p. 3.

[40] G. M. Weiss, K. Yoneda, and T. Hayajneh, "Smartphone and smartwatch-based biometrics using activities of daily living," *IEEE Access*, vol. 7, pp. 133 190–133 202, 2019.

[41] S. Mekruksavanich and A. Jitpattanakul, "Deep learning approaches for continuous authentication based on activity patterns using mobile sensing," *Sensors*, vol. 21, no. 22, p. 7519, 2021.

[42] R. San-Segundo, R. Cordoba, J. Ferreiros, and L. F. D'Haro-Enríquez, "Frequency features and gmm-ubm approach for gait-based person identification using smartphone inertial signals," *Pattern Recognition Letters*, vol. 73, pp. 60–67, 2016.

[43] G. Stragapede, R. Vera-Rodriguez, R. Tolosana, A. Morales, J. Fierrez, J. Ortega-Garcia, S. Rasnayaka, S. Seneviratne, V. Dissanayake, J. Liebers *et al.*, "Ijcb 2022 mobile behavioral biometrics competition (mobileb2c)," in *2022 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 2022, pp. 1–7.