Adaptive Deep Feature Fusion for Continuous Authentication with Data Augmentation

Yantao Li, Li Liu, Huafeng Qin, Shaojiang Deng, Mounim A. El-Yacoubi, and Gang Zhou

Abstract—Mobile devices are becoming increasingly popular and are playing significant roles in our daily lives. Insufficient security and weak protection mechanisms, however, cause serious privacy leakage of the unattended devices. To fully protect mobile device privacy, we propose ADFFDA, a novel mobile continuous authentication system using an Adaptive Deep Feature Fusion scheme for effective feature representation, and a transformer-based GAN for Data Augmentation, by leveraging smartphone built-in sensors of the accelerometer, gyroscope and magnetometer. Given the normalized sensor data, ADFFDA utilizes the transformer-based GAN consisting of a transformer-based generator and a CNN-based discriminator to augment the training data for CNN training. With the augmented data and the especially designed CNN based on the ghost module and ghost bottleneck, ADFFDA extracts deep features from the three sensors by the trained CNN, and exploits an adaptive-weighted concatenation method to adaptively fuse the CNN-extracted features. Based on the fused features, ADFFDA authenticates users by using the one-class SVM (OC-SVM) classifier. We evaluate the authentication performance of ADFFDA in terms of the efficiency of the transformer-based GAN, GAN-based data augmentation, CNN architecture, adaptive-weighted feature fusion, and OC-SVM classifier. The experimental results show that ADFFDA obtains the best authentication performance w.r.t representative approaches, by achieving a mean equal error rate of 0.01%.

Index Terms—Continuous authentication, deep feature fusion, adaptive weights, data augmentation, CNN, OC-SVM

1 Introduction

In the digital age of inter-connectivity, mobile devices, such as smartphones, smartwatches, and tablets, have become increasingly popular for their mobility and convenience. With the rapid increase of memory and computational sources, these mobile devices play significant roles in our daily lives as they allow people to save multimedia data, perform social or communication activities, and interact with user-related IoT devices. More and more private and sensitive information, as a result, is stored on or transmitted between mobile devices. For these reasons, it becomes imperative to devise smart technologies to prevent privacy leakage from people's mobile devices. Due to the importance of the privacy protection, primary security methods, namely knowledge-based and physiological biometrics-based mechanisms, have been widely applied in mobile devices. As the knowledge-based mechanisms, such as PINs and graphical patterns, highly depend on users' knowledge, they suffer from several adversarial attacks, such as guessing [1] and eye glimpsing [2]. On the other side, since the physiological biometrics-based mechanisms require physiological traits as input, such as face and fingerprints, they suffer from direct and indirect attacks [3], chief among them, replaying [4] and spoofing [5]. Therefore, stronger protection mechanisms are required for information security on mobile devices.

The continuous authentication is an implicit process of identifying users that relies on capturing their behavioral attributes by leveraging the resources and sensors of mobile devices [6]. The continuous authentication mechanisms can frequently authenticate mobile device users via behavioral biometrics-based approaches, based on unique behavioral patterns of users, such as touch gestures [7], [8], [9], gaits [10], [11], [12], heartbeats [13], [14], and chest motion [15]. These behavioral biometrics-based systems help to secure mobile banking, shopping, remote meeting, and so on, because these applications require to continuously validate the user's identity during an entire session. In this respect, some works utilizing deep learning methods to extract deep features have achieved a relatively high accuracy [16], [17], [18]. These works, however, face the challenge of insufficient training data. Although the authors in [16] provide five data augmentation approaches, i.e., permutation, sampling, scaling, cropping, and jittering (widely used in image augmentation) for sensor data augmentation and achieve approximately 4.66% median EER with a dataset size of 200, these geometric transformation-based data augmentation techniques are difficult to capture the deep information inside the real data. The created data lack, therefore, diversity

- *Y. Li, L. Liu, and S. Deng are with the College of Computer Science, Chongqing University, Chongqing 400044, China.*
  E-mail: {yantaoli,20162969,sj_deng}@cqu.edu.cn
- *H. Qin is with the School of Computer Science and Information Engineering, Chongqing Technology and Business University, Chongqing 400067, China.*
  E-mail: qinhuafengfeng@163.com
- *M. A. El-Yacoubi is with SAMOVAR, Telecom SudParis, CNRS, Institut Polytechnique de Paris, 91120 Palaiseau, France.*
  E-mail: mounim.el_yacoubi@telecom-sudparis.eu
- *G. Zhou is with the Department of Computer Science, William & Mary, Williamsburg, VA, 23185, USA.*
  E-mail: gzhou@cs.wm.edu

Manuscript received November XX, 2021; revised XXXX XX, 202X. This work was supported in part by the National Natural Science Foundation of China under Grants 62072061 and 61976030, in part by the Fundamental Research Funds for the Central Universities under Grant 2021CDJQY-026, in part by the Funds for Creative Research Groups of Chongqing Municipal Education Commission under Grant CXQT21034, in part by Chongqing Talent Program under Grant CQYC201903246, and in part by the Scientific and Technological Research Program of Chongqing Municipal Education Commission under Grants KJQN201900848 and KJZD-K201901105. (Corresponding author: Huafeng Qin.)

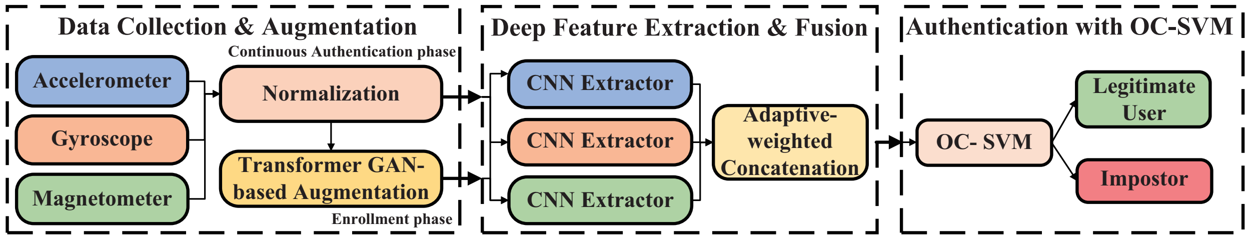
Data Collection & Augmentation
Continuous Authentication phase
Accelerometer
Gyroscope
Magnetometer
Normalization
Transformer GAN-based Augmentation
Enrollment phase
Deep Feature Extraction & Fusion
CNN Extractor
CNN Extractor
CNN Extractor
Adaptive-weighted Concatenation
Authentication with OC-SVM
OC- SVM
Legitimate User
Impostor

Fig. 1: Architecture of ADFFDA

related to the high dependence characterizing the original data [19], [20]. Generative adversarial networks (GANs) using CNN as backbones, namely, CNN-based GANs, can learn information about data probability distribution and generate realistic-like data from the real data distribution. They have been exploited in computer vision tasks, such as image synthesis [21], image enhancement [22] and image editing [23], to improve the training of deep learning models. Although CNN-based GAN data augmentation approaches have been widely used in the image processing field, the extracted features are local for each layer and have the global receptive field only in the last few layers causing the loss of feature resolution and fine details. For example, Li *et al.* use a conditional Wasserstein generative adversarial network consisting of a CNN-based generator and a CNN-based discriminator for data augmentation, resulting in a partial improvement of authentication accuracy [24]. Moreover, existing CNN-based GAN data augmentation approaches face the challenge of the way they can be adapted for time-series data augmentation. The emergence of the transformers address these limitations thanks to their strong representation capabilities and their general and simple architecture that is suitable for smart devices [25]. On the other hand, since a single biometric modality suffers from significant limitations associated with the inherent variability of biometric traits, due to noise, and poor data quality, the combination of multiple biometric sources is a known strategy to improve identification accuracy. In this regard, some works dedicated to feature fusion strategies have managed to improve accuracy [26], [27], but they are facing the challenge of inefficient feature combination representation. For instance, Zhang *et al.* utilize deep feature fusion strategy of a weighted concatenation to fuse iris and periocular modalities for mobile identification enhancement [29]. Concretely, this work and other similar ones are usually based on fixed weight combination schemes, such as direct concatenation [28], and weighted concatenation [29], which hinder their full improvement potential. Efficient feature fusion strategies for continuous authentication with high feature combination are therefore another challenge to be addressed.

In order to tackle the aforementioned two challenges of insufficient training data and inefficient feature combination, we are among the first to explore a transformer-based GAN to generate additional data for CNN training and design an adaptive-weighted concatenation method for CNN-extracted feature fusion. In this paper, we propose ADFFDA, a novel and mobile continuous authentication system using an Adaptive Deep Feature Fusion for effective feature combination and a transformer-based GAN for Data Augmentation, which leverages smartphone built-in sensors, i.e. accelerometer, gyroscope and magnetometer, to capture users' behavioral patterns. Specifically, the operation of ADFFDA is composed of the enrollment phase and the continuous authentication phase. In the enrollment phase, ADFFDA utilizes a transformer-based GAN to augment the normalized training data for the designed CNN training, extracts CNN-based features, then implements a new adaptive-weighted concatenation scheme to fuse deep features, and finally trains the one-class SVM (OC-SVM), thereby learning to discriminate a legitimate user from an impostor. In the continuous authentication phase, with the trained CNN, feature fusion model, and OC-SVM classifier, ADFFDA continuously authenticates the current user as the legitimate user or an impostor by relying on the testing data from the accelerometer, gyroscope and magnetometer, thereby allowing the owner to operate the smartphone as long as he/she is authenticated.

The main contributions of this work can be summarized as follows:

- We propose ADFFDA, a novel and mobile continuous authentication system using an adaptive deep feature fusion for effective feature combination, and a transformer-based GAN for data augmentation. ADFFDA comprises five modules: data collection, data augmentation, deep feature extraction, deep feature fusion, and authentication with OC-SVM.
- We are among the first to explore a transformer-based GAN composed of a transformer-based generator and a CNN-based discriminator in order to generate additional training data for CNN training. Based on the ghost module and ghost bottleneck, we especially design a CNN to learn discriminative features for adaptive deep feature fusion.
- We design an adaptive-weighted concatenation method for CNN-extracted feature fusion, which adaptively assigns weights to the three features associated with the accelerometer, gyroscope and magnetometer, in order to ensure feature representation.
- We conduct extensive experiments to evaluate the authentication performance of ADFFDA. The experimental results indicate that ADFFDA attains the best authentication performance, when comparing with representative state-of-the-art approaches, reaching a mean EER of 0.01% .

The rest of this work is organized as follows: Section 2 provides ADFFDA including the system model, adversary model, and overview. In Section 3, we describe our methods of collecting and preprocessing the sensor data, respectively. Section 4 details the transformer-based GAN for CNN training data augmentation. In Section 5, we elaborate on how to extract deep features by the designed CNN. Section 6 describes an adaptive deep feature fusion strategy for effective feature representation. In Section 7, we introduce the OC-SVM classifier for the authentication. Section 8 evaluates the authentication performance of ADFFDA. In Section 9, we review the existing literature about behavioral biometrics, data augmentation, and feature fusion in authentication systems. Section 10 concludes this work.

2 ADFFDA

In this section, we begin with the system model, then describe the adversary model, and finally present the overview of ADFFDA.

2.1 System Model

We consider a mobile continuous authentication system that utilizes users' behavioral patterns extracted from smartphone built-in sensors, i.e., the accelerometer, gyroscope and magnetometer, to continuously authenticate the user as a legitimate user or an imposer once the user starts operating smartphones (e.g., finger touch, gesture, and wrist motion). The authentication process of our system typically consists of two phases: the enrollment phase for data augmentation and model training, and the continuous authentication phase for user authentication with the trained models. In the enrollment phase, user data are first augmented and then utilized to train a feature extractor. The trained feature extractor extracts deep features, which are then fused by a feature fusion strategy. Based on the trained feature extractor and fusion model, the legitimate user registers his/her profile through interaction on smartphones, and a classifier is then trained. In the continuous authentication phase, sensors start collecting real-time behavioral data, when users operate the smartphone integrating the trained feature extractor, fusion model, and classifier. Deep features from the collected data are extracted by the extractor and then are fused by the fusion model. The trained classifier identifies the user as a legitimate user or an impostor. For example, the system can identify a user when he/she texts a message on an unlocked smartphone as a legitimate user or impostor. If the user is authenticated as an impostor, the smartphone will be immediately locked; otherwise, it will allow the legitimate user to continue using the phone.

2.2 Adversary Model

Continuous authentication is an implicit process of validating a legitimate user based on capturing behavioral patterns by leveraging resources and built-in sensors on mobile devices. We consider a user spoofing attack against such a continuous authentication system, namely a mimic attack. In this attack, an adversary first observes the way a legitimate user conducts gestures/touches on his/her phone to pass the authentication, and then practices to mimic the user's behaviors for conducting the attack.

2.3 ADFFDA Overview

The key idea of ADFFDA is to continuously authenticate smartphone users using a transformer-based GAN for data augmentation and an adaptive-weighted concatenation method for CNN-extracted feature fusion. Fig. 1 describes the system architecture of ADFFDA, which is composed of two phases: the enrollment phase and the continuous authentication phase. Concretely, in the enrollment phase, ADFFDA learns a profile of a legitimate user by utilizing the training data to train the transformer-based GAN, CNN, feature fusion model, and the OC-SVM classifier. After the legitimate user's profile is learned, ADFFDA enters the continuous authentication phase to identify users by exploiting the trained CNN, feature fusion model, and the classifier on testing data.

As illustrated in Fig. 1, ADFFDA consists of five modules: data collection, data augmentation, deep feature extraction, deep feature fusion, and authentication with OC-SVM. Specifically, the data collection module exploits three smartphone built-in sensors, i.e. the accelerometer, the gyroscope, and the magnetometer, to sample users' fine-grained and coarse-grained behavioral data. The data augmentation module is composed of data preprocesing and transformer GAN-based augmentation. It first normalizes the three sensor data respectively, and then utilizes a transformer-based GAN consisting of a transformer-based generator and a CNN-based discriminator to create additional training data for deep feature extraction. Then, the feature extraction module uses a CNN based on the ghost module and ghost bottleneck to learn discriminative deep features for the three sensors, respectively. With the three CNN-extracted feature vectors, the feature fusion module exploits an adaptive-weighted concatenation method to adaptively assign weights to the three feature vectors, and then concatenates the three weighted vectors to one fused feature vector. Based on the adaptively concatenated deep features, the OC-SVM classifier is trained to learn the legitimate user's profile based on the training data. With the trained OC-SVM and CNN, the authentication module identifies, on the testing data, the current user as a legitimate user (the owner) or an imposter. ADFFDA allows the continuous usage of the smartphone if the user is classified as a legitimate user; otherwise, it requires the user's initial identification.

3 Data Collection and Preprocessing

In this section, we first explain how to collect the data for ADFFDA and then describe how to preprocess the collected data for the transformer-based GAN and CNN training.

3.1 Data Collection

In ADFFDA, the sensor data of an operation are collected by using a mobile smartphone equipped with the accelerometer, gyroscope and magnetometer sensors. The accelerometer and gyroscope motion sensors capture a user's coarse-grained patterns such as arm movements and gaits, and fine-grained motion patterns such as touch gestures, respectively. The magnetometer position sensor determines the smartphone's physical position in the real frame of reference, which is leveraged for data calibration to acquire more precise motion information. Each collected sample can be represented as $(x_a, y_a, z_a, x_g, y_g, z_g, x_m, y_m, z_m)^T \in \mathbb{R}^9$, where x, y, z represent the three axes of a sensor, and a, g, m indicate the three accelerometer, gyroscope, and magnetometer sensors, respectively.

3.2 Data Preprocessing

For a time period t, n $(n = t \times f)$ samples of raw accelerometer, gyroscope and magnetometer data can be collected, where f indicates the sensor sampling rate. Each sensor data can be represented by a $3 \times n$ matrix: $D_a = [\mathbf{d_x^a}, \mathbf{d_y^a}, \mathbf{d_z^a}]^T = \begin{bmatrix} x_a^1 & x_a^2 & \cdots & x_a^n \\ y_a^1 & y_a^2 & \cdots & y_a^n \\ z_a^1 & z_a^2 & \cdots & z_a^n \end{bmatrix}$, for the accelerometer for example, and then all the three sensor data can be expressed as $D = [D_a, D_g, D_m]$.

For transformer-based GAN training, CNN training and fusion model training, we normalize each axis of each sensor. Specifically, we normalize each axis i of each sensor s into $(0, 1)$ by $\mathbf{d_i^s}_{norm} = \frac{\mathbf{d_i^s} - min(\mathbf{d_i^s})}{max(\mathbf{d_i^s}) - min(\mathbf{d_i^s})}$, where $i = x, y, z$ for the three axes of each sensor and $s = a, g, m$ for the three sensors, respectively. Then $D_{anorm} = [\mathbf{d^a_x}_{norm}, \mathbf{d^a_y}_{norm}, \mathbf{d^a_z}_{norm}]^T$ for normalized sensor data of the accelerometer can be obtained, and then all the normalized sensor data can be described as $D_{norm} = [D_{anorm}, D_{gnorm}, D_{mnorm}]$. Note that we do not remove the gravity influence on data since it affects all the sensors.

4 Transformer GAN-based Data Augmentation

Proposed by I. Goodfellow *et al.*, generative adversarial networks (GANs) are commonly composed of a generator G that captures the data distribution and a discriminator D that estimates the probability that a sample comes from the training data rather than G [30]. Most of GANs use convolutional neural networks (CNNs) as GAN backbones, such as DCGAN (deep convolutional GAN) [31], FCCGAN (fully connected and convolutional GAN) [32], SinGAN (generative model from a single natural image) [33], and BiGAN (bidirectional GAN) [34]. CNN-based GANs have achieved high performance for computer vision tasks, such as image synthesis [21], image enhancement [22] and image editing [23]. CNN-based GAN data augmentation has been widely used in the image processing field, as the latter benefits from the effectiveness of the convolutional layer's effective feature extraction and the network structure that starts with low-resolution images and then progressively increases the resolution by adding layers [35]. However, the extracted features are local from each layer and have the global receptive field only in the last few layers, which is not suitable for time-series data. Actually, to process variable length sequences of inputs, not only CNN but even RNN (Recurrent Neural Network) using their internal state (memory) encounter hard-training issues, training gradient explosion or vanishing, and training without parallelization. The emergence of the transformer breaks through the above limitations, which allows the modeling of dependencies without regard to their temporal distance in the input or output sequences. The transformer is able to draw global dependency between input and output, and allows significant parallellization [25]. The authors in [36] built a GAN completely free of convolutions for natural images and achieved new state-of-the-art results. However, the use of a transformer-based architecture to augment time-series data is still unexplored. Next, we exploit the transformer architecture to construct a time-series data augmentation network.

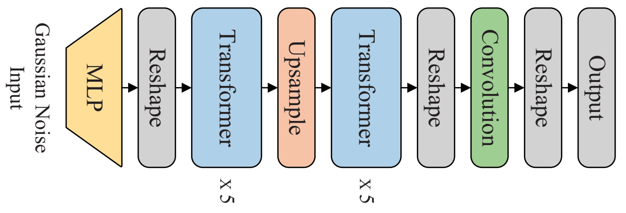

Fig. 2: Architecture of Generator

TABLE 1: Generator

Operator	Output	Repeat	Multi-head
Random Noise	200	1	-
MLP	1600	1	-
Reshape	50 × 32	1	-
Transformer	50 × 32	5	2
Upsample	200 × 8	1	-
Transformer	200 × 8	5	2
Reshape	20 × 10 × 8	1	-
Conv	20 × 10 × 3	1	-
Reshape	200 × 3	1	-

4.1 Transformer-based GAN

As our behavioral data are time series and because of the reasons above, we have chosen the transformer encoder as the basic block for our GAN [25]. The transformer encoder consists of a multi-head self-attention module and a feed-forward multiple-layer perceptron (MLP) with GELU nonlinearity.

4.1.1 Generator

The transformer-based generator is composed of 2 stages, each of which stacks 5 transformer encoder blocks using 2 heads for the multi-head attention mechanism, as shown in Fig. 2 and Table 1. Since sensor data are sequential, the positional encoding is added before each transformer encoder, thereby generating samples that fit time series of real samples. In order to better fit the sensor data with a long time window while reducing the number of parameters, an upsample block is inserted between the two transformer stages, which is composed of a reshaping and a pixel-shuffle modules. After the two stages of transformer encoder, 1D (one-dimensional) sequence data are generated, where the embedding dimension reduces to a quarter of the original one. To generate sample data with the same embedding dimensions as sensor data, 1D sequence data are first reshaped to a 2D feature map, and then the embedding dimension of the map is compressed by Conv, and finally is reshaped to 1D sequence.

4.1.2 Discriminator

The discriminator can be regarded as a classifier that identifies the inputs are real data or generated data. Since a

TABLE 2: Discriminator

Operator	Output	# Kernel	KSize	Padding	Stride
Sensor Data	$1 \times 200 \times 3$	-	-	-	-
Conv (LeakyReLU)	$32 \times 100 \times 3$	32	(5,1)	(2,0)	(2,1)
Conv (LeakyReLU)	$64 \times 50 \times 3$	64	(5,1)	(2,0)	(2,1)
Conv (LeakyReLU)	$128 \times 25 \times 3$	128	(5,1)	(2,0)	(2,1)
AvgPool	128	-	-	-	-
FC (LeakyReLU)	512	-	-	-	-
FC	1	-	-	-	-

transformer-based discriminator seems to be an inferior "competitor" and is much more data-hungry, a CNN-based discriminator can alleviate this issue [36]. Thus, we design the discriminator by a CNN architecture, consisting of three convolutional layers (Conv), one average pooling layer (AvgPool) and two fully connected layers (FCs), respectively, as illustrated in Table 2. The activation function, a leaky version of the Rectified Linear Unit (LeakyReLU), is applied on the three Convs and the first FC, and the loss function is WGAN-GP [37]. For the three Conv layers, we use a kernel with a fixed size, where the number of kernels is doubled to make the feature map small and thick, and the second element of the kernel size is set to 1 to ensure that data features corresponding to different axes can be extracted respectively. The padding and stride ensure data length can be reduced by 50% after each convolutional layer. The outputs are then fed to AvgPool to obtain 1D feature sequences and finally are fed to two FC layers to classify the inputs into two classes, real samples or fake samples.

4.2 Data Augmentation

On the one hand, the generator takes a random noise with format of (200×1) as its input (to train the correlation between 200 samples) and then feeds the noise to a MLP to obtain a vector with length of $L \times C$, where L represents the length of a time window and C indicates C-dimensional embedding for sensor data ($L = 50$ and $C = 32$, by default). Then the reshaped sensor axis embedding with format of $(L \times C)$ are fed into the first transformer encoder and the correspondence between different embeddings, which is users' behavior trajectory in different time points, is calculated recursively. The upsample block first reshapes the 1D sequence data back to a 2D feature map with format of $(H \times W \times C)$, where $H \times W = L$. It subsequently adopts the pixel-shuffle to obtain an upsampling feature map with format of $(2H \times 2W \times C/4)$ and then combines the first two elements in the embedding dimension resulting in a 1D sequence with format of $(4L \times C/4)$, thereby extending the length of the generated sequence. The 1D sequence data of $(4L \times C/4)$ are then fed to the second transformer encoder and the correspondence is recursively calculated as well. Afterwards, the output of the second transformer encoder is reshaped to a 2D feature map of $(4L/10 \times 10 \times C/4)$ where the feature map channel is equal to 1D sequence's embedding dimension, and then we apply Conv to further compress the channel number to 3 which corresponds to the three sensor axes. Finally, the Reshape layer reshapes the 2D feature map back to 1D sequence to generate an output of (200×3), which corresponds to 2-second D_{norm} data.

On the other hand, the 2-second D_{norm} data combined with the produced data by the generator with shape of $(1 \times 200 \times 3)$ are fed into the discriminator. In the first convolutional layer, there are 32 filters with kernel size of (5,1). The filters for the second and third Conv layers are 64 and 128, respectively. All the three Conv layers have the same kernel size of $(5, 1)$, padding $(2, 0)$ and stride $(2, 1)$. They continuously extract features from different axes and compress the length of data sequence from 200 to 100, then 50, and finally to 25. The feature map of $(128 \times 25 \times 3)$ is produced after the three Convolutions, then is fed to the AvgPool layer to obtain features of (128×1), and finally two FCs are applied to obtain a scalar for the classification of the produced data.

Next, we use the WGAN-GP loss function to calculate the distance between the 2-second D_{norm} data distribution and the created data distribution, and then use backpropogation to further update the generator parameters after updating discriminator parameters every n times until a fixed number of epochs is reached, where n is a hyperparameter that can balance the generator training process and the discriminator training process.

Finally, the data generated by the transformer-based GAN can be described as: D_{tran}, where $|D_{tran}| = 3 \times |D_{norm}|$ (6-second data). After the data augmentation, the dataset can be expressed as: $D_{dataset} = [D_{norm}, D_{tran}]$, which will be used for deep feature extraction.

5 Deep Feature Extraction

In this section, we first design the CNN architecture based on the ghost module and ghost bottleneck. Then, we elaborate on how to extract deep features by the designed CNN.

5.1 CNN Design

Traditional CNNs usually need lots of parameters and floating point operations to achieve a satisfactory accuracy. Thus, lightweight and efficient network architectures with acceptable performance, such as MobileNet [38] and ShuffleNet [39], have been investigated for mobile devices with fewer parameters and calculations. Inspired by MobileNetV2 [40] that can project features to a low-dimensional representation with a linear convolution and ghostNet [41] that can fully reveal information underlying intrinsic features, we design a CNN architecture based on the ghost module and ghost bottleneck.

5.1.1 Ghost module

The ghost module utilizes a few filters to generate more feature maps from one standard convolutional layer [41]. Instead of a standard convolution, the ghost module splits a standard convolutional layer into two parts: 1) a standard convolution with fewer filters that generates a few intrinsic feature maps, and 2) depthwise convolution on previous intrinsic feature maps to gain more feature maps. The structure of the ghost module is described in Fig. 3 and Table 3. As we can see, the ghost module comprises a Conv layer, a depthwise convolution (DWConv) layer and a concatenation operation. Specifically, the $K \times 1$ Conv layer with a Rectified Linear Unit (ReLU) takes a $C \times H \times W$ input and produces a $(N/s) \times H \times W$ intrinsic feature map, where C is the input channel number, H and W indicate the height and

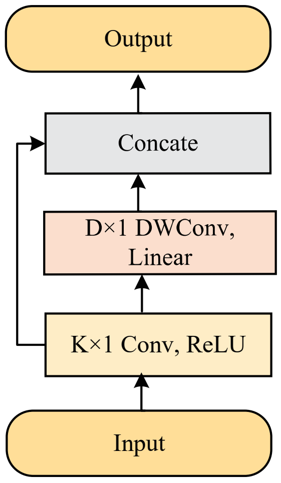

Fig. 3: Ghost Module

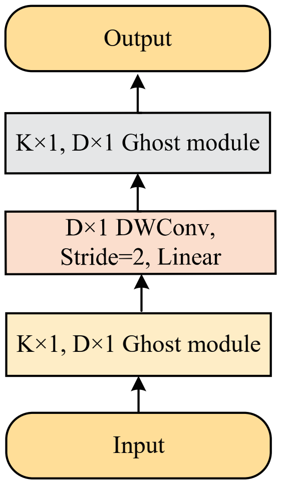

Fig. 4: Ghost Bottleneck

width of the input data, s is a hyper-parameter to control the number of generated intrinsic feature maps, and N/s is the number of convolutional kernels, respectively. Then, the $D \times 1$ DWConv layer applies a linear operation on the produced feature map to generate a $(s-1) \times (N/s) \times H \times W$ ghost feature map, where D is the kernel size of the DW-Conv. Finally, the ghost module outputs a $N \times H \times W$ feature map by concatenating the intrinsic feature map and the ghost feature map. The computational cost of the ghost module is $(N/s) \times H \times W \times C \times K \times 1 + (s-1) \times (N/s) \times H \times W \times D \times 1$. However, with the same input and kernel size, the computation cost for a standard convolution is $N \times H \times W \times C \times K \times 1$. Compared with a standard convolution, the computation cost of the ghost module is $\frac{(N/s)\times H\times W\times C\times K\times 1+(s-1)\times(N/s)\times H\times W\times D\times 1}{(N/s)\times H\times W\times C\times K\times 1}$ of that of the standard convolution, which can be simplified as $\frac{(N/s)\times C\times K+(N/s)\times D}{N\times C\times K} = \frac{C+(s-1)}{s\times C} = \frac{1}{s}$, since D is close to K and s is much less than C. We set $s = 2$ in our experimental setting, and thus the computational cost of the ghost module is just half of the standard one but the performance remains.

5.1.2 Ghost bottleneck

The structure of the ghost module is described in Fig. 4 and Table 4. As shown, the ghost bottleneck consists of a ghost module, a DWConv layer, and another ghost module. Specifically, the first ghost module expends a $C \times H \times W$ feature map in a low-dimensional space to a $tC \times H \times W$ output in a high-dimensional space, where t is an expansion factor. The DWConv layer with kernel size $D \times 1$ and stride of 2 downsamples the generated feature map to produce a $tC \times (H/2) \times W$ output. The second ghost module maps the input manifold in a high-dimensional space into a $C \times (H/2) \times W$ output in a low-dimensional space, which has the same channel number to the input of the first ghost module. In addition, the kernel sizes of $K \times 1$ and $D \times 1$ in ghost module indicate the kernel sizes of the Conv layer and the DWConv layer, respectively.

5.1.3 CNN architecture

Building on the ghost module and ghost bottleneck, we design a CNN architecture to extract deep features, as illustrated in Fig. 5. As shown in Fig. 5, the designed CNN architecture comprises a Conv layer, three ghost bottleneck layers, an AvgPool layer, and two FC layers.

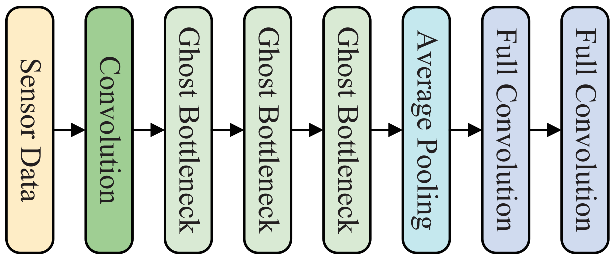

Fig. 5: Architecture of CNN

5.2 CNN-based Feature Extraction

Based on the designed CNN, we present a deep feature extraction approach to learn discriminative features for adaptive deep feature fusion. Given the transformer GAN-augmented data $D_{dataset}$, we exploit the designed CNN architecture to extract deep features from the raw accelerometer, gyroscope and magnetometer data, respectively. For each of the three sensors, there are 600 samples (2 seconds $\times$ 100 $Hz\times$ 3 axes $\times$ 1 sensor) for a 2-second time window, which are reshaped as $1\times 200\times 3$. As demonstrated in Table 5, with a $1 \times 200 \times 3$ sensor data input, the Conv layer with 32 filters, kernel size 9×1 and stride 2 produces a $32 \times 100 \times 3$ output. Three stages of the ghost bottlenecks gradually decrease the sizes of their input feature maps by using different kernel sizes of $(7\times 1), (5\times 1), (3\times 1)$. That is, the height H of the feature map decreases, but the width C corresponding to the three axes of a sensor and the channel number C remain the same. All the ghost bottlenecks are applied with 32 filters, expansion ratio $t = 6$ and stride $= 2$. The AvgPool layer outputs a preliminary 32-dimensional feature vector F_i for a sensor $i(i = \{a, g, m\})$. After the three feature vectors corresponding to the three sensors are extracted, they are fed to a feature fusion module to generate more discriminative deep features for final classification. In addition, the two FC layers are utilized to classify the users.

6 Adaptive Deep Feature Fusion

Since different sensors are available on smart devices, it is imperative to optimally exploit the data provided by them to identify users. While different sensors contribute differently to the authentication, it is necessary to use a feature fusion strategy to jointly learn the features from different sensors to accurately and robustly authenticate users. However, most of the existing feature fusion strategies are based on fixed combination weights, including direct concatenation [28], and weighted concatenation [29]. In our work, we propose an adaptive deep feature fusion strategy (adaptive-weighted concatenation method) to optimize the joint feature representation of the accelerometer, gyroscope, and magnetometer sensors, by adaptively assigning weights to the different feature vectors. Finally, we concatenate the latter into a fused feature vector for accurate and robust classification.

TABLE 3: Ghost Module

Input	Operator	Output
$C \times H \times W$	$K \times 1$ Conv, Padding, ReLU	$(N/s) \times H \times W$
$(N/s) \times H \times W$	$D \times 1$ DWConv, Padding, Linear	$(s-1) \times (N/s) \times H \times W$
$(N/s) \times H \times W + (s-1) \times (N/s) \times H \times W$	Concate	$N \times H \times W$

TABLE 4: Ghost Bottleneck

Input	Operator	Output
$C \times H \times W$	$K \times 1$, $D \times 1$ Ghost module	$tC \times H \times W$
$tC \times H \times W$	$D \times 1$ DWConv, Stride=2, Linear	$tC \times (H/2) \times W$
$tC \times (H/2) \times W$	$K \times 1$, $D \times 1$ Ghost module	$C \times (H/2) \times W$

TABLE 5: CNN Architecture

Operator	Output	t	# Kernel	KSize	Stride
Sensor Data	$1 \times 200 \times 3$	-	-	-	-
Convolution	$32 \times 100 \times 3$	-	32	(9,1)	(2,1)
Ghost Bottleneck	$32 \times 50 \times 3$	6	32	(7,1)	(2,1)
Ghost Bottleneck	$32 \times 25 \times 3$	6	32	(5,1)	(2,1)
Ghost Bottleneck	$32 \times 13 \times 3$	6	32	(3,1)	(2,1)
Average Pooling	32	-	-	-	-
Full Connection	512	-	-	-	-
Full Connection	44	-	-	-	-

Given the three 32-dimensional feature vectors F_a, F_g, and F_m extracted by the designed CNN, the adaptive deep feature fusion (ADFF) strategy adaptively fuses the three sensor feature vectors to a 96-dimensional feature vector $ADFF$, where $ADFF = [ADFF_a, ADFF_g, ADFF_m]$. Since the fusion processes for the three 32-dimensional feature vectors F_a, F_g, and F_m are the same, we take the accelerometer feature vector F_a as an example to elaborate the fusion process of the adaptive-weighted sum, as demonstrated in Fig. 6. As illustrated in Fig. 6, the ADFF strategy adaptively assigns weights according to F_a for the three feature vectors, and then adds the three weighted feature vectors together to generate a 32-dimensional feature vector $ADFF_a$. Specifically, for each feature vector F_i ($i = \{a, g, m\}$), ADFF maps F_i to three feature vectors Q_i (query), K_i (key) and an identity F_i, where $Q_i = F_i \times W_{Q_i}$ and $K_i = F_i \times W_{K_i}$ ($i = \{a, g, m\}$). For example, $Q_a = F_a \times W_{Q_a}$ and $K_a = F_a \times W_{K_a}$, where "$\times$" indicates the dot product. Then, ADFF calculates the dot products of Q_a with all keys K_i ($i = \{a, g, m\}$) to generate three corresponding scores S_a, S_g, S_m for the three sensor features. Next, ADFF applies a softmax function on the three scores S_i ($i = \{a, g, m\}$) to obtain the corresponding weights for the three identities. Finally, ADFF computes the dot products of the identities with their weights and sums them together to obtain the weighted feature vector $ADFF_a$. In summary, the 32-dimensional weighted feature vector $ADFF_a$ is generated by Eq. (1):

$$ADFF_a = \sum_{i=\{a,g,m\}} softmax\left(Q_a K_i^T\right) \times F_i \quad (1)$$

where $Q_i = F_i \times W_{Q_i}$ and $K_i = F_i \times W_{K_i}$, for $i = \{a, g, m\}$. Similarly, the corresponding weighted feature vectors $ADFF_g$ and $ADFF_m$ are obtained by Eqs. (2) and (3), respectively:

$$ADFF_g = \sum_{i=\{a,g,m\}} softmax\left(Q_g K_i^T\right) \times F_i \quad (2)$$

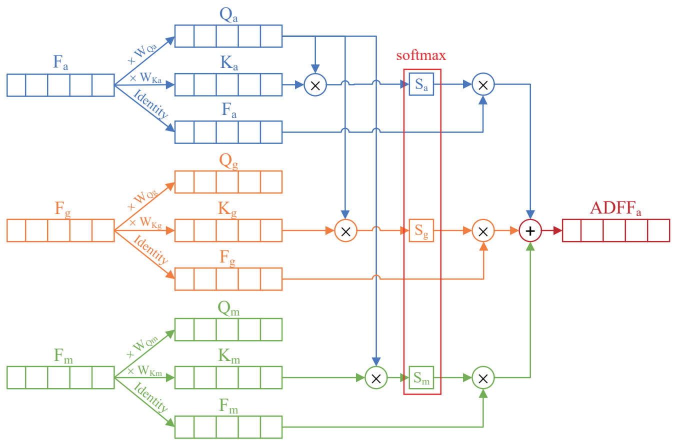

Fig. 6: Adaptive-weighted Sum for Accelerometer

and

$$ADFF_m = \sum_{i=\{a,g,m\}} softmax\left(Q_m K_i^T\right) \times F_i \quad (3)$$

Finally, as demonstrated in Fig. 7, the 96-dimensional fused feature vector $ADFF$ is obtained by concatenation of the three weighted feature vectors: $ADFF = [ADFF_a, ADFF_g, ADFF_m]$. Note that our feature fusion model (ADFF) is trained in the enrollment phase and has 0.07M parameters, which is a matrix computation in the continuous authentication phase. Due to the low number of parameters and matrix-level computation complexity, ADFF strategy is suitable for resource-limited mobile devices.

7 Authentication with OC-SVM

Based on the fused deep features of $ADFF$, ADFFDA utilizes a one-class support vector machine (OC-SVM) classifier to authenticate users. OC-SVM maps data points into a high-dimensional feature space with the kernel function and searches a surface of a minimal hyper-sphere that contains

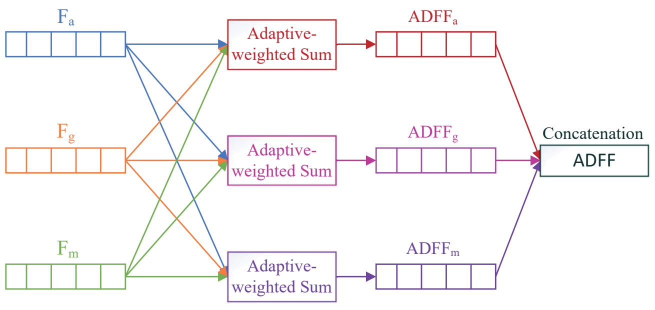

Fig. 7: Adaptive-weighted Concatenation

as many objective data points as possible [42]. Data points within the hyper-sphere are regarded as positive samples while data points outside of the hyper-sphere are recognized as negative samples.

In the enrollment phase, the OC-SVM is trained from the positive samples, based on the adaptive-weighted deep features with the radial basis function kernel and ADFFDA learns the profile of the legitimate user from the training data. In the authentication phase, the trained OC-SVM maps the testing data into the same high-dimensional feature space as in the training phase's and ADFFDA classifies the current user as a legitimate user or an impostor.

8 Performance Evaluation

In this section, we introduce the experimental settings and describe the extensive experiments that we have conducted to evaluate the performance of ADFFDA. For performance evaluation, we begin with the efficiency of transformer-based GAN and the effectiveness of GAN-based data augmentation approach. Then, we evaluate the efficiency of the CNN architecture, and the effectiveness of our adaptive-weighted feature fusion scheme. Finally, we assess the efficiency of the OC-SVM classifier.

8.1 Experimental Settings

In this section, we depict the experimental settings involving the dataset collection, the training processes of the transformer-based GAN, CNN, feature fusion model, and OC-SVM classifier, respectively, and the evaluation metrics.

8.1.1 Dataset

To collect the sensor data, we developed a data collection tool for Android phones to collect behavioral data from the users' interaction with the phones. After receiving the IRB approval from William & Mary in 2014, we started recruiting volunteers for data collection. In this regard, we recruited 100 volunteers (53 male and 47 female) to operate the phones equipped with the developed tool. To obtain high-quality data, the volunteers were required to conduct three designed tasks: (1) document reading, (2) text production, and (3) navigation on a map to locate a destination. Once they logged in the developed tool, a reading, writing, or a map navigation session was randomly assigned, each lasting about 5 to 15 minutes. Based on the assignments, they were expected to perform 24 sessions including 8 reading sessions, 8 writing sessions, and 8 map navigation sessions to collect totally 2 to 6 hours of behavior traits [43]. We recorded the accelerometer, gyroscope and magnetometer sensor readings with a sampling rate of $f = 100\ Hz$ as CSV files on the phones.

In our experiments, based on the collected data from the accelerometer, gyroscope and magnetometer sensors, we have retained the data from 88 volunteers in CSV files on the phones and have chosen the first 100-minute samples for each volunteer with a 2-second window size as the experimental dataset. Note that we find the data of 24 sessions are incomplete for the rest 12 volunteers due to the insensitive sensors in one of the experimental phones.

8.1.2 Training

Based on the 88 volunteers' dataset, 44 volunteers' samples were randomly selected for transformer-based GAN training and the corresponding augmented samples were used for CNN training, 34 for the deep feature fusion model training, and 10 for classifier training. The reasons are that 1) CNN as the feature extractor needs more training data (44 volunteers') to enhance feature generalization; and 2) 10 volunteers' data are convenient for ten-fold cross-validation of the OC-SVM classifier. Specifically, we first train the transformer-based GAN on 44 volunteers' samples with WGAN-GP loss function and Adam optimizer to obtain a generator for each volunteer until 2000 epochs. Then, with the augmented data generated by the trained generators, we train a CNN based on the cross-entropy loss function and RMSprop optimizer until 200 epochs for the accelerometer, gyroscope and magnetometer sensors, respectively. Next, based on the CNN-extracted features from the 34 volunteers' samples, we train the feature fusion model by the same process to the CNN until 10 epochs. Finally, we utilize the ten-fold cross-validation to train the OC-SVM classifier 100 times on the features extracted from the trained CNN and fusion model from 10 volunteers' samples.

For *Transformer-based GAN training*, we have trained three independent transformer-based GANs for the three accelerometer, gyroscope and magnetometer sensors, respectively, on the corresponding 44 volunteers' samples with a batch size of 512. For each GAN, with a batch of 200-dimensional Gaussian noise as inputs, the generator creates a batch data. The created data and the real data are fed to the discriminator to learn to discriminate them from each other, and the parameters of the generator and discriminator are updated accordingly. For the GAN trained on the accelerometer data or the gyroscope data, we use WGAN-GP loss function and Adam optimizer to update the learning rate for 2000 epochs with an initial value of 0.000015 or 0.00001 for both the generator and discriminator, where we train the generator once every 5 epochs of the training of the discriminator. For the GAN trained on the magnetometer data, we also use Adam optimizer to update the learning rate for 2000 epochs with an initial value of 0.000015 for both the generator and discriminator, where we train the generator once every 2 epochs of the training of the discriminator. After the transformer-based GAN is trained, only generators remain for generating additional sensor data.

For *CNN training*, based on the transformer-GAN augmented data from 44 volunteers' samples, 90% volunteers' samples are randomly selected for CNN training and the rest 10% are used for testing, with a batch of size 256. For each epoch, a batch of training samples are fed into the designed CNN, and then the outputs with user labels are exploited to compute cross-entropy loss. For optimization, RMSprop is used to update the network parameters. We set the initial learning rate as 0.001 and the weight decay as 0.0001. The total number of training epochs is 200. We train three CNNs for the accelerometer, gyroscope and magnetometer sensors, respectively.

For *feature fusion model training*, we utilize the trained CNNs with fixed parameters to extract features from the three sensors' data. With the 34 volunteers' samples, 90% volunteers' samples are randomly selected for fusion model training, and the rest 10% are used for testing, with a batch of size 256. For each epoch, with the CNN-extracted features, we explore the same training process as CNN to train the fusion model with the cross-entropy loss function and RMSprop optimizer until 10 epochs.

For *Classifier training*, we train the OC-SVM classifier with RBF as the kernel function, for which the kernel coefficient $\gamma = 0.005$, and an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors are set as 0.01. Based on the 10 volunteers, we randomly select one as a legitimate user and the rest 9 as impostors. The positive samples from the legitimate user are evenly divided into 10 subsets, where each of the subsets is sequentially utilized as a testing dataset and the remaining 9 subsets are exploited as a training dataset. In order to keep the same amount of positive samples, 1/9 the samples randomly selected from the 9 impostors are used as a negative training dataset. The selected negative samples are shuffled and then evenly divided into 10 subsets, each of which is sequentially exploited as a testing dataset. With the 10 volunteers' samples, we utilize the ten-fold cross-validation to train the OC-SVM classifier 100 times, and the average of the 100 results is used as the classification result.

8.1.3 *Evaluation Metrics*

We utilize the equal error rate (EER) as the main evaluation criterion, and consider the false acceptance rate (FAR), false rejection rate (FRR), accuracy, and F1 score as the complementary metrics for the performance evaluation of ADFFDA. Given the four basic metrics of true positive (TP), true negative (TN), false positive (FP), and false negative (FN), we define the FAR, FRR, and EER, where $FAR = \frac{FP}{FP+TN}$ indicating the probability that an impostor is falsely identified as a legitimate user [44], $FRR = \frac{FN}{FN+TP}$ representing the probability that a legitimate user is incorrectly classified as an impostor [6], and EER is the point where the FAR equals to the FRR [9]. We define the accuracy as $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, which indicates the capability of distinguishing between a legitimate user and an impostor. Given $precision = \frac{TP}{TP+FP}$ indicating the capability of correctly classifying a user as a legitimate user, and $recall = \frac{TP}{TP+FN}$ representing the capability of classifying a legitimate user as the legitimate user, F1 score can be defined as $F1\ score = \frac{2 \times precision \times recall}{precision + recall}$, which is the harmonic mean of the precision and recall indicating the comprehensive performance of the proposed system [45].

8.2 Efficiency of Transformer-based GAN

To evaluate the efficiency of the proposed transformer-based GAN for augmenting sensor data, we exploit three metrics, i.e. discriminator loss, maximum mean discrepancy (MMD) and t-distributed stochastic neighbor embedding (t-SNE), to measure the quality of the generated sensor-like data by the designed transformer-based GAN. Discriminator loss represents the Earth-Mover distance between the real sensor data and the generated sensor-like data until the network converges [46]. The higher the quality of the generated data, the closer the loss is to 0. MMD measures the distance between the distributions of the real sensor data and the generated data [47]; the higher the quality of the generated data, the closer the MMD is to 0. t-SNE maps the high-dimensional generated data non-linearly into two-dimensions, where the corresponding data can be visualized [48].

Based on the transformer-based GAN training process in Sec. 8.1.2, we visualize the discriminator losses for the three sensors, i.e. the accelerometer, gyroscope, and magnetometer, in Fig. 8. As shown in Fig. 8, the three discriminator losses sharply drop until 500 epochs, then gradually increase, and finally slightly oscillate around a small value (close to 0) along with the increase of the training epochs, where they reach 0 around 1100 epochs. The discriminator losses indicate that the three sensor generated data have high similarity to real data. Moreover, with the trained transformer-based GAN, we calculate the MMD of 50 real and 50 generated samples for one epoch until 2000 epochs for the three sensors, respectively, as shown in Fig. 9. As illustrated in Fig. 9, the MMDs of the accelerometer and magnetometer data rapidly decrease until 500 epochs and then slightly oscillate around a small value (close to 0), while the MMD of the gyroscope data slowly decreases and finally converges to a small value. The MMD results indicate that the generated sensor data show high qualities. In addition, the distributions of 500 real and 500 generated samples of the three sensors are visualized by t-SNE in Fig. 10. In Fig. 10, we exploit the blue point, orange star, and green cross to represent the accelerometer, gyroscope, and magnetometer data, respectively, where the light color indicates the real sensor data and dark color represents the generated sensor data. As illustrated in Fig. 10, the sensor data are clearly separated by colors into three clusters corresponding to three sensors. In each cluster, the real data and generated data are closely distributed together, which indicates the generated sensor data have high similarity to the real sensor data.

8.3 Effectiveness of Transformer GAN-based Data Augmentation

To evaluate the effectiveness of the transformer GAN-based data augmentation approach, we conduct comparison between ADFFDA with transformer GAN-based data augmentation and ADFFDA without data augmentation on different dataset sizes. With a data size varying from 100 to 700, we plot boxes of EERs for ADFFDA with the transformer GAN-based data augmentation approach (orange box plot) and ADFFDA without augmentation approach (blue box plot), as demonstrated in Fig. 11. As shown in Fig. 11, EERs

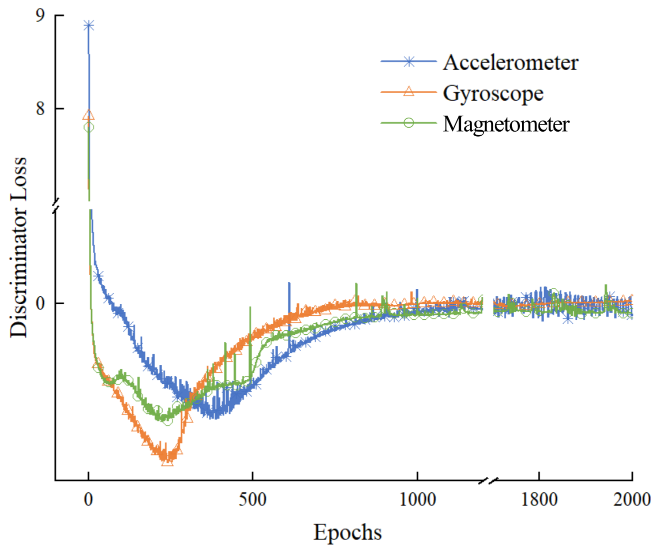

Fig. 8: Discriminator Loss

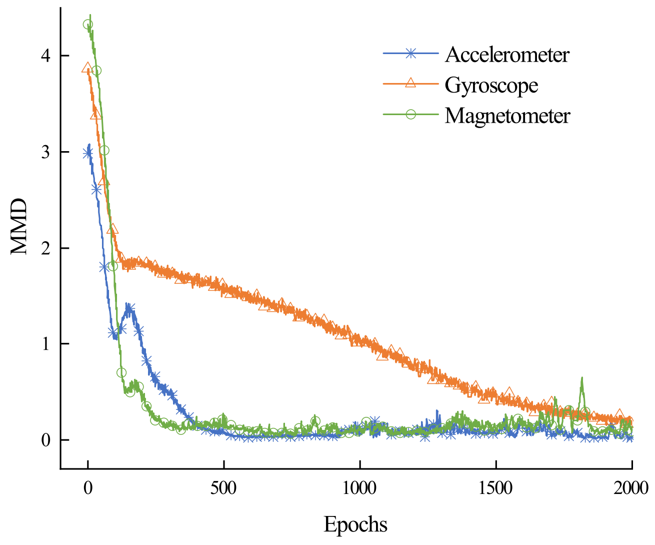

Fig. 9: MMD

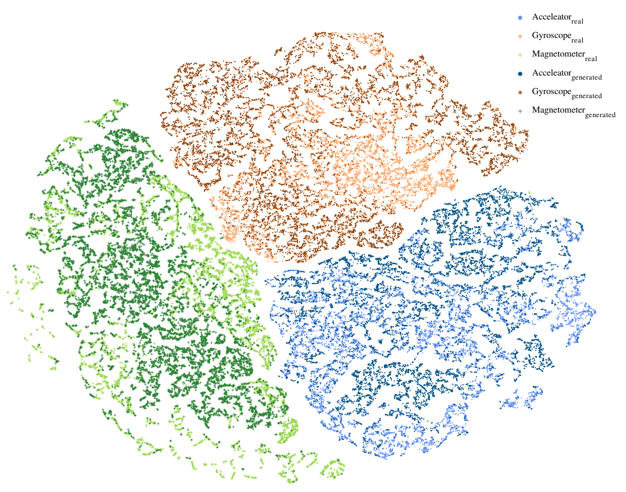

Fig. 10: Distribution of real and generated samples by t-SNE

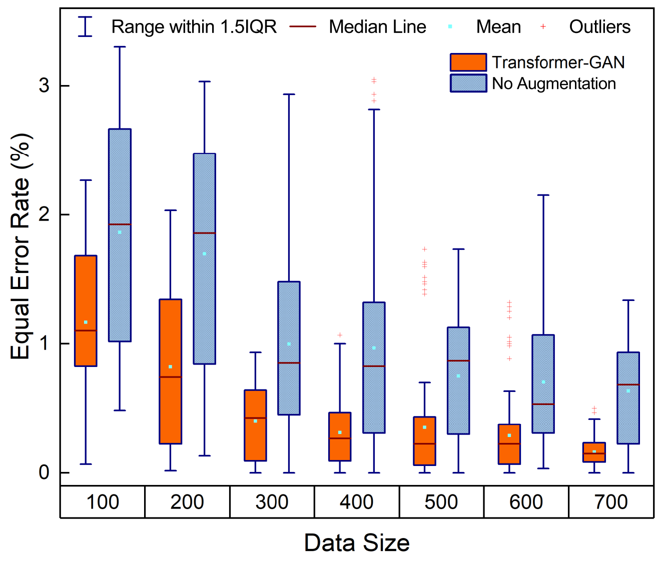

Fig. 11: EER of ADFFDA with or without Transformer-based GAN augmentation over different data sizes

for both the approaches gradually decrease as the data size increases. Moreover, EERs with the transformer GAN-based data augmentation approach are associated, for all the data sizes, with clearly lower values compared to those obtained with no data augmentation. That is to say, the transformer GAN-based data augmentation approach significantly improves the identification accuracy of ADFFDA. In addition, we list, in the first two rows of Table 6, the mean EERs with or without transformer GAN-based data augmentation over different data sizes. As shown, ADFFDA with transformer GAN-based data augmentation achieves an EER of 1.62% with data size of 700, which dramatically improves that without data augmentation. From data size of 500 onwards, the margin of improvement though data augmentation is at least 3.99%.

To further assess the effectiveness of the transformer GAN-based data augmentation scheme, we compare ADFFDA with representative data augmentation approaches, i.e. CWGAN [24], permutation, sampling, scaling, cropping and jittering [16]. Based on our data, we conduct the same experiment by replacing the transformer-based GAN in ADFFDA with the data augmentation techniques mentioned above, and the corresponding results are illustrated in Fig. 12 and tabulated in Table 6. As described in Fig. 12, EERs of all the augmentation approaches gradually decrease, in general, with the increase of the data size, which validates the efficiency of data augmentation. As depicted in Table 6, CWGAN reaches the best EER of 2.76% with data size of 600; Jittering and permutation achieve the lowest EERs of 2.94% and 6.96% on data size of 500, respectively; Sampling, scaling, and cropping obtain the best EERs of 8.52%, 6.42%, and 4.97% on data size of 700, respectively. In comparison, our transformer-based GAN reaches the lowest EER of 1.62% and standard deviation (SD) of 1.15% and significantly outperforms the representative data augmentation approaches above, by margins of 1.14% (2.76%, CWGAN with data size of 600) and 0.97% (2.12%, jittering with data size of 500) at least.

TABLE 6: Mean (SD) EER (%) with different data augmentation approaches over different data sizes.

Approach \ Data size	100	200	300	400	500	600	700
No Augmentation	18.65 (8.47)	16.98 (8.46)	9.98 (7.48)	9.65 (7.93)	7.51 (4.88)	7.03 (5.31)	**6.36 (3.82)**
Transformer-GAN	11.64 (5.80)	8.20 (5.90)	4.03 (2.72)	3.13 (2.61)	3.52 (4.30)	2.89 (3.15)	**1.62 (1.15)**
CWGAN	5.95 (5.94)	4.20 (4.13)	4.25 (3.61)	2.97 (3.33)	6.19 (5.77)	**2.76 (2.62)**	3.54 (3.22)
Permutation	19.35 (6.53)	17.88 (6.88)	13.03 (5.09)	12.90 (5.40)	**6.96 (3.88)**	8.59 (4.39)	7.69 (3.99)
Sampling	13.73 (7.32)	10.15 (6.39)	9.42 (5.32)	13.52 (7.45)	11.72 (7.20)	12.65 (7.33)	**8.52 (4.51)**
Scaling	20.78 (7.78)	16.51 (6.18)	10.23 (5.30)	8.72 (4.43)	8.52 (4.20)	8.50 (5.35)	**6.42 (3.31)**
Cropping	22.79 (7.57)	17.68 (6.90)	9.84 (4.41)	8.53 (4.40)	5.50 (3.28)	5.71 (3.75)	**4.97 (2.92)**
Jittering	14.40 (6.39)	15.03 (8.56)	12.42 (7.50)	11.38 (7.71)	**2.94 (2.12)**	3.68 (2.82)	3.77 (2.45)

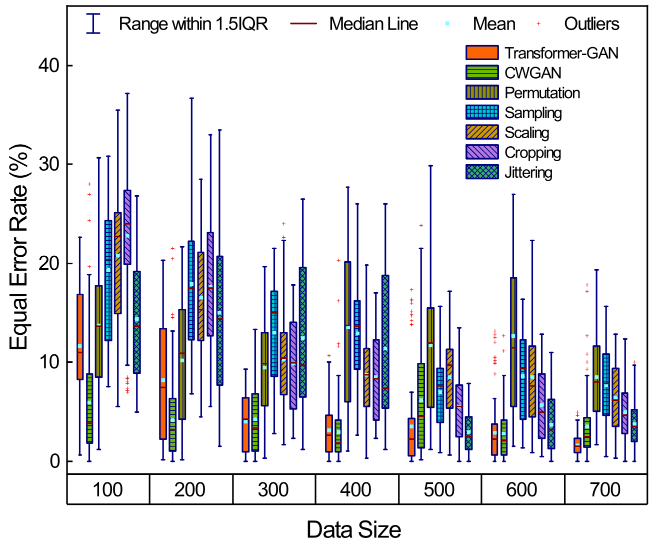

Fig. 12: EER with different data augmentation approaches over different data sizes

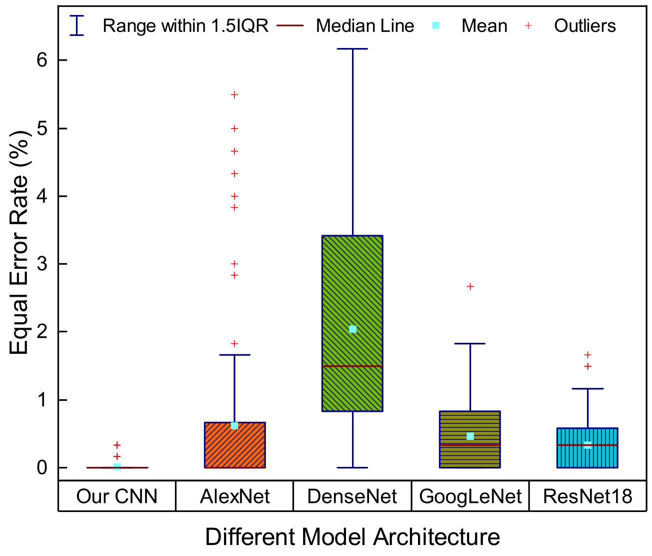

Fig. 13: EER of representative model architectures

TABLE 7: Mean EER, SD (%) and parameter on representative network architectures

Model	EER	SD	Parameter
Ours	0.01	0.06	0.14M
AlexNet	0.61	1.16	1.24M
DenseNet	2.04	1.70	2.36M
GoogLeNet	0.46	0.49	3.41M
ResNet18	0.33	0.37	3.93M

8.4 Efficiency of CNN

To evaluate the efficiency of the designed CNN architecture, we compare our CNN to existing representative model architectures, including AlexNet [49], DenseNet [50], GoogLeNet [51], and ResNet [52]. To adapt these model architectures to our CNN input format ($1 \times 200 \times 3$), we set the second dimension of the kernel size as 1 for all the convolution layers and pooling layers, and the second dimension of padding as 0. Moreover, in order to output the same feature dimension in the intermediate layer as our CNN, we add two FC layers with 32 and 1024 nodes before the fully connected layer used for the final classification used to obtain 32-dimensional features that provide sufficient information for subsequent classification.

Based on our data, we conduct the same experiment by replacing our CNN in ADFFDA with AlexNet, DenseNet, GoogLeNet, and ResNet18. We plot the EER of the different architectures in Fig. 13. As illustrated in Fig. 13, our CNN achieves the best EER w.r.t other model architectures. Furthermore, we list the EER, SD, and number of parameters of all the network architectures in Table 7. As listed in Table 7, compared to the other model architectures, our CNN reaches the lowest EER of 0.01% with the smallest SD of 0.06%, while having the lowest number of parameters, i.e. 0.14 million. ResNet18 is the second best, but its performance is far beyond ours. Overall, although AlexNet, DenseNet, GoogLeNet, and ResNet18 have shown excellent performance in computer vision tasks, they reach much lower authentication accuracy for sensor data. The reason may be that: 1) these deep learning models are originally designed for image processing, and thus not suitable for sensor data; and 2) these complex models should be trained by a large amount of data.

The main conclusion in this experiment, is that among all the representative model architectures, our CNN is by large the most suitable network architecture for ADFFDA.

8.5 Effectiveness of Adaptive-weighted Feature Fusion

To evaluate the effectiveness of our fusion strategy, we compare our adaptive-weighted concatenation fusion scheme with existing strategies, including serial fusion (feature direct concatenation), parallel fusion [28], and deep feature fusion [29]. Based on our dataset, we conduct the same experiment by replacing our strategy with the three existing strategies above in ADFFDA. We plot the accuracy and F1 score for the different fusion strategies in Fig. 14 and list the mean EER, accuracy and F1 score in Table 8. As illustrated in Fig. 14, our adaptive-weighted concatenation

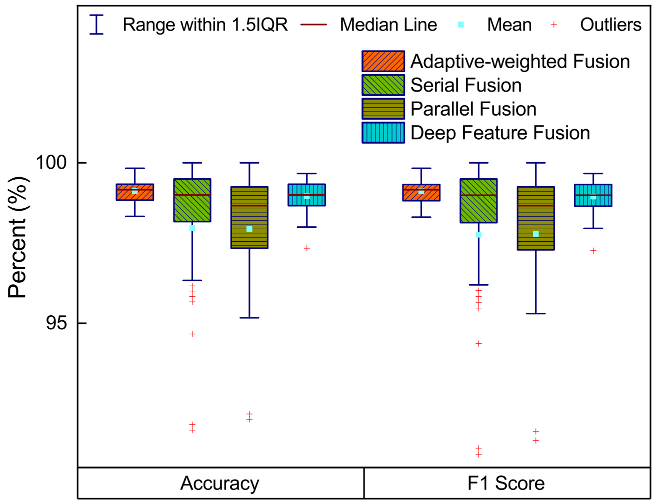

Fig. 14: Comparison of different fusion strategies

TABLE 8: Mean (SD) EER, accuracy and F1 score (%) of different fusion strategies

Fusion strategy	EER	Accuracy	F1 score
Ours	0.01 (0.06)	99.12 (0.37)	99.11 (0.37)
Serial	0.07 (0.34)	97.97 (3.50)	97.76 (4.47)
Parallel	1.42 (1.71)	97.93 (3.32)	97.78 (4.58)
Deep Fusion	0.03 (0.09)	98.96 (0.41)	98.95 (0.43)

fusion scheme shows the highest accuracy and F1 score, along with the lowest standard deviation (SD), compared to the three existing fusion strategies. Moreover, as listed in Table 8, our strategy reaches a 0.01% EER, a 99.12% accuracy, and a 99.11% F1 score, which dramatically surpasses the deep feature fusion strategy with margins of 0.02%, 0.16%, and 0.16%, respectively. This shows the power of our adaptive-weighted feature fusion scheme as an effective fusion strategy for ADFFDA.

Furthermore, we visualize the adaptive weight assignment for the three sensors, i.e. accelerometer, gyroscope, and magnetometer, by randomly selecting three samples from the 10 testing users, as illustrated in Fig. 15. Fig. 15 shows three weight assignments S_a, S_g, and S_m for each of the 10 users (x axis) corresponding to three weighted feature vectors $ADFF_a$, $ADFF_g$, and $ADFF_m$ for three samples (y axis). Specifically, for user 6, for example, S_g dominates the weight assignments in the weighted feature vectors $ADFF_a$, $ADFF_g$, and $ADFF_m$ in sample 1, which indicates that the gyroscope contributes the most to the feature fusion compared to the other two sensors. In sample 2, however, the three weights of S_a, S_g, and S_m evenly contribute to the weighted feature vectors, which indicates all the sensor features make even contributions to the fused feature. In sample 3, S_a contributes the least since it is assigned the least weights. From Fig. 15, we can find that when the same user operates the smartphone in different ways, the contributions of the sensor features to the feature fusion vary accordingly. For different users, the contributions of the sensor features are generally distinct. This illustrates how effective our adaptive-weight feature fusion scheme is for ADFFDA.

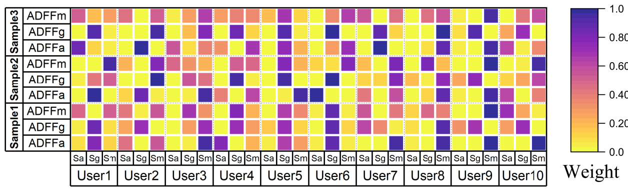

Fig. 15: Visualization of adaptive weight assignment

TABLE 9: Mean (SD) EER, FAR, and FRR (%) of different classifiers

Classifier	EER	FAR	FRR
OC-SVM	0.01 (0.06)	0.00 (0.00)	1.76 (0.74)
LOF	0.05 (0.13)	0.00 (0.03)	0.58 (0.47)
IF	0.25 (0.32)	0.00 (0.00)	4.93 (1.76)
kNN	0.33 (0.46)	0.66 (0.93)	0.00 (0.00)

8.6 Efficiency of OC-SVM classifier

To investigate the efficiency of our OC-SVM classifier, we compare it with representative classifiers, namely the local outlier factor (LOF), isolation forest (IF), and k-nearest neighbors (kNN). Specifically, LOF first measures the local deviation of the data point to its neighbors, and then decides whether a data point is an outlier using the local density estimated by kNN based on a given distance metric [53]. IF detects abnormal data points by subsampling the data set to construct iTrees and further integrate multiple iTrees into a forest to detect abnormal data [54]. kNN takes every new observation and locates the observation in feature space with respect to all training observations [55]. We conduct the same experiment by replacing the OC-SVM classifier with LOF, IF, and kNN in ADFFDA. Based on our dataset, we plot the EER of different classifiers, as shown in Fig. 16. As described in Fig. 16, OC-SVM and LOF classifiers show better EERs than those obtained with the IF and kNN classifiers, and OC-SVM has less outliers compared to LOF. Moreover, we tabulate the mean EER, FAR, and FRR with SDs of different classifiers in Table 9. As shown in Table 9, the OC-SVM classifier has by far the best EER i.e. 0.01% and FAR i.e. 0.00%, and dramatically surpasses the LOF, as the OC-SVM's EER is 5 times less than LOF's (0.01% vs 0.05%) classier of 0.04% EER. In addition, according to the experiments, the training time for OC-SVM is 0.20 s while LOF 's is 19.25 s. Therefore, we conclude, overall, that the OC-SVM classifier is by large the most suitable classifier for ADFFDA.

To further assess the performance of the OC-SVM classifier, we evaluate the accuracy on the 10 volunteers' data and 34 volunteers' data for testing in the feature fusion model training process. Specifically, based on n ($n = 10, 20, 30, 40$) volunteers, we randomly select one as a legitimate user and the rest $n - 1$ as impostors to conduct ten-fold cross-validation on the OC-SVM classifier. Table 10 depicts the mean EER with SD for the OC-SVM classifier on different number of users. As listed in Table 10, with the increase of the users, the EER gradually increases and remains at 0.06% with 30 or 40 users, which indicates that the OC-SVM classifier still shows a high accuracy on more users' data.

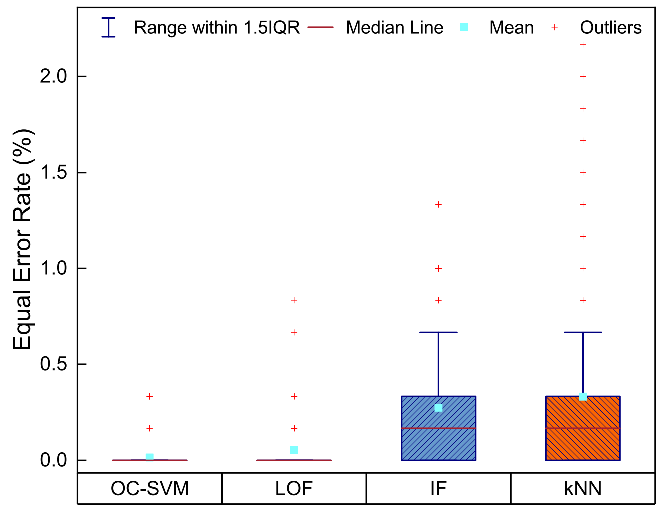

Fig. 16: Comparison of different classifiers

TABLE 10: Mean (SD) EER (%) of the OC-SVM classifier with Different User Number

User Number	10	20	30	40
EER	0.01 (0.06)	0.04 (0.15)	0.06 (0.21)	0.06 (0.21)

8.7 Security Analysis

To evaluate the performance of ADFFDA in user authentication against spoofing, we have conducted experiments under mimic attacks on 10 volunteers. In the mimic attack, we expect an adversary to try his/her best to mimic the behavioral patterns of a legitimate user. For mimic attack data, we add a random Gaussian noise (range from 0 to 0.1) to the 10 volunteers' original data, which indicates very tiny vibration in every sensor. We utilize the ten-fold cross-validation to train the OC-SVM classifier, where 50% legitimate user's data and 50% corresponding mimic data are used as the testing dataset. The experimental results show that ADFFDA obtains a 0.01% FAR and 0.16% EER, which indicate that ADFFDA can resist mimic attacks.

9 Literature Review

In this section, we review the existing literature on behavioral biometrics-based methods, data augmentation approaches as well as feature fusion strategies, respectively, in authentication systems.

9.1 Behavioral Biometrics in Authentication Systems

In order to explore effective behavioral biometrics for different authentication systems, several modalities, such as touch gestures, gaits, heartbeats, and chest motions, have been exploited as unique characteristics for behavioral biometrics-based authentication. The authors in [7] explored six types of touch gestures (single-tap, swipe forward, swipe backward, swipe down, two-finger swipe forward, and two-finger swipe backward) to authenticate users in a continuous and noninvasive authentication system for wearable glasses. In [8], the authors exploited physical characters of touching fingers by investigating active vibration signal transmission through fingers in a behavior-irrelevant on-touch user authentication system. The authors in [9] utilized a touch sensor to analyze the on-screen gesture while using an inertial sensor to analyze the device's motion caused by the touch gesture for user authentication. In [10], the authors leveraged unique gait patterns derived from acceleration readings in mobile healthcare systems to detect possible user spoofing attacks. The authors in [11] used a smartphone-based accelerometer to capture gait data continuously in the background for a smartphone-based gait authentication system. In [12], the authors proposed an agile algorithm to extract gait-specific features in a Wi-Fi-based human identification system. The authors in [13] leveraged Photoplethysmography (PPG) sensors in wrist-worn wearables to extract individual characteristics of clean heartbeat signals from PPG signals for a novel mobile two-factor authentication system. In [14], the authors utilized the inherent correlation between sounds and chest motion caused by deep breathing for heart sound-based authentication. The authors in [15] leveraged the distinctive chest motions during speaking to establish a secure multi-factor authentication system.

Although these behavioral biometrics are effective and efficient on specific wireless devices, we exploit the accelerometer, gyroscope and magnetometer sensors built-in modern phones to capture users' behavioral patterns implicitly. Compared with these behavioral biometrics-based authentication systems, ADFFDA generates additional sensor data to strengthen CNN training and fuses data from multiple sensing modalities to enhance feature representation, thereby significantly improving the authentication performance with a mean EER of 0.01% (Table 9).

9.2 Data Augmentation in Authentication Systems

In order to address the limited training data issue for authentication systems, data augmentation approaches have been exploited for generating additional training data to improve the authentication accuracy. Specifically, the authors in [56] utilized a variety of data augmentation techniques including shift up+zoom, mirror, blur, equalize histogram, darken, brighten, inject noise for periocular image augmentation in a CNN-based periocular authentication system. In [57], the authors exploited the augmentation techniques of permutation, scaling, and jittering to produce four different deformations from an original signal sample, increasing the number of samples of each training user in a continuous authentication through gait analysis. The authors in [16] utilized five data augmentation approaches, i.e. permutation, sampling, scaling, cropping and jittering, to generate additional sensor data in a smartphone-based continuous authentication system. In [17], the authors augmented behavioral biometric features of HMOG including hand movement, orientation, and grasp with tap characteristics in continuous smartphone authentication, which considerably improved authentication performance. The authors in [58] transformed the landmark pixel coordinates in one camera to any new camera pose to create synthesized training samples for an acoustics and vision based authentication system. In [59], the authors used three data augmentation approaches, i.e. the conditional Wasserstein GAN, selective VAE, and selective WGAN to augment EEG data for emotion recognition enhancement. The authors in [18] exploited a multiscale and multidirection GAN (MSMDGAN) for data augmentation in a CNN-based single palm-vein identification. In [24], the authors used a conditional Wasserstein GAN (CWGAN) consisting of a CNN-based generator and a CNN-based discriminator for data augmentation in a smartphone-based continuous authentication system.

Different from the aforementioned data augmentation approaches in authentication systems, we are the first to exploit a transformer based-GAN composed of a transformer-based generator and a CNN-based discriminator to generate sensor data for a mobile continuous authentication system. In comparison with these data augmentation based authentication systems, ADFFDA shows the superiority by reaching the lowest EER of 1.62% on data size of 700 (Table 6).

9.3 Feature Fusion in Authentication Systems

In order to improve the feature efficiency in authentication systems, feature fusion strategies have been utilized to enhance the feature representation. Concretely, the authors in [60] fused user data from the keyboard, mouse, and graphical user interface interactions, which resulted in a more accurate authentication result according to a broader view of the user's computer activity in a behavioral biometric system. In [61], the authors concatenated the gait and keystroke feature vectors that were normalized by the Min–Max normalization technique to a single feature vector in a continuous smartphone authentication method. In [28], the authors utilized two feature fusion strategies, serial and parallel, to concatenate the selected features from the accelerometer, gyroscope, and magnetometer sensors in a continuous smartphone authentication system. The authors in [26] exploited a local feature fusion to the ECG and finger veins using an updated version of canonical correlation analysis in a multimodal authentication system. In [29], the authors developed a deep feature fusion network that applied maxout units into CNNS to generate a compact representation for each modality and then concatenated the discriminative features of iris and periocular with weights. The authors in [27] utilized a balanced feature concatenation strategy to fuse a fixed-number deep features from the accelerometer and gyroscope on smartphones in a CNN-based continuous authentication system.

Although the aforementioned feature fusion strategies do improve authentication systems, we differ in that we propose an adaptive-weighted concatenation method to fuse CNN-extracted features from the accelerometer, gyroscope and magnetometer sensors, which has led to a much more effective feature representation, as shown by our experiments. Compared with these feature fusion based authentication systems, ADFFDA surpasses them with the highest accuracy of 99.12% (Table 8).

10 Conclusion

In this paper, we have presented ADFFDA, a novel and mobile continuous authentication system using an adaptive deep feature fusion scheme for effective feature representation and a transformer-based GAN for data augmentation, leveraging the accelerometer, gyroscope and magnetometer sensors on smartphones. When a user operates on smartphones, ADFFDA utilizes the designed CNN to extract deep features from the normalized sensor data. It then employs an adaptive-weighted concatenation scheme to fuse the deep features. Finally, it authenticates the user by the trained OC-SVM classifier. We have evaluated ADFFDA through extensive experiments and the experimental results show that ADFFDA is able to authenticate users efficiently, and obtains, by far, the lowest EER w.r.t representative works of the state-of-the-art. ADFFDA achieves the superior performance, it, however, highly relies on user behaviors, such as finger touch, gesture, and wrist motion, in the training phase. If new behaviors occur in the testing phase, ADFFDA will not work well.

References

[1] T. V. Nguyen, N. Sae-Bae and N. Memon, "Draw-a-pin: Authentication using finger-drawn pin on touch devices," *Comput. Secur.*, vol. 66, pp. 115–128, 2017.

[2] W. Meng, L. Zhu, W. Li, J. Han and Y. Li, "Enhancing the security of fintech applications with map-based graphical password authentication," *Future Gener. Comput. Syst.*, vol. 101, pp. 1018–1027, 2019.

[3] R. Jain and C. Kant, "Attacks on Biometric Systems: An Overview," *Int. J. Adv. Sci. Res.*, vol. 1, no. 07, pp. 283-288.

[4] N. Erdogmus and Sebastien Marcel, "Spoofing face recognition with 3D masks," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 7, pp. 1084–1097, 2014.

[5] A. Roy, N. Memon and A. Ross, "Masterprint: Exploring the vulnerability of partial fingerprint-based authentication systems," *IEEE Trans. Inf. Forensics Secur.* vol. 12, no. 9, 2013–2025, 2017.

[6] M. Abuhamad, A. Abusnaina, D. Nyang and D. Mohaisen, "Sensor-based continuous authentication of smartphones' users using behavioral biometrics: A contemporary survey," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 65–84, 2021.

[7] G. Peng, G. Zhou, D. T. Nguyen, X. Qi, Q. Yang and S. Wang, "Continuous authentication with touch behavioral biometrics and voice on wearable glasses," *IEEE Trans. Hum. Mach. Syst.*, vol. 47, no. 3, pp. 404–416, 2017.

[8] X. Xu, J. Yu, Y. Chen, Q. Hua, Y. Zhu, Y.-C. Chen and M. Li, "TouchPass: towards behavior-irrelevant on-touch user authentication on smartphones leveraging vibrations," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Sep. 2020, pp. 1-13.

[9] X. Zhang, Y, Yin, L. Xie, H. Zhang, Z. Ge, S. Lu, "TouchID: User Authentication on Mobile Devices via Inertial-Touch Gesture Analysis," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 4, Article 162, Dec. 2020.

[10] Y. Ren, Y. Chen, M. C. Chuah and J. Yang, "Smartphone based user verification leveraging gait recognition for mobile healthcare systems," In *2013 IEEE Int. Conf. Sensing, Commu. Netw. (SECON)*, Oct. 2013, pp. 149-157.

[11] M. Muaaz and R. Mayrhofer, "Smartphone-based gait recognition: From authentication to imitation," *IEEE Trans. Mob. Comput.*, vol. 16, no. 11. pp. 3209–3221, 2017.

[12] Y. Zhang, Y. Zheng, G. Zhang, K. Qian, C. Qian and Z. Yang, "GaitSense: Towards Ubiquitous Gait-Based Human Identification with Wi-Fi," *ACM Trans. Sen. Netw.* vol. 18, no. 1, Article 1, 24 pages, Sep. 2021.

[13] Y. Cao, Q. Zhang, F. Li, S. Yang and Y. Wang, "PPGPass: Nonintrusive and Secure Mobile Two-Factor Authentication via Wearables," *Proc. IEEE Conf. Computer Communications (INFOCOM)*, pp. 1917-1926, Jul. 2020.

[14] C. Huang, H. Chen, L. Yang and Q. Zhang, "BreathLive: Liveness Detection for Heart Sound Authentication with Deep Breathing," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 1, Article 12, 25 pages, Mar. 2018.

[15] Y. Chen, M. Xue, J. Zhang, Q. Guan, Z. Wang, Q. Zhang, and W. Wang, "ChestLive: Fortifying Voice-based Authentication with Chest Motion Biometric on Smart Devices," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 4, Article 148, 25 pages, Dec. 2021.

[16] Y. Li, H. Hu and G. Zhou, "Using Data Augmentation in Continuous Authentication on Smartphones," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 628-640, Feb. 2019.

[17] Z. Sitová et al., "HMOG: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 877–892, 2016.

[18] H. Qin, M. A. El-Yacoubi, Y. Li and C. Liu, "Multi-Scale and Multi-Direction GAN for CNN-Based Single Palm-Vein Identification," *IEEE Trans. Inf. Forensis Security*, vol. 16, pp. 2652-2666, 2021.

[19] D.A.V. Dyk, X.L. Meng, "The art of data augmentation," *J. Comput. Graph. Stat.*, vol. 10, no. 1, pp. 1–50, 2001.

[20] S. Dieleman, K. W. Willett and J. Dambre, "Rotation-invariant convolutional neural networks for galaxy morphology prediction," *Mon. Notices Royal Astron. Soc.*, vol. 450, no.2, pp. 1441-1459, 2015.

[21] F. Zhan, H. Zhu, S. Lu, "Spatial Fusion GAN for Image Synthesis," In Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 3653–3662.

[22] Y. Jiang et al., "EnlightenGAN: Deep Light Enhancement Without Paired Supervision," *IEEE Trans. Image Process.*, vol. 30, pp. 2340-2349, 2021.

[23] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," In Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2018, pp. 5505–5514.

[24] Y. Li, J. Luo, S. Deng and G. Zhou, "CNN-based Continuous Authentication on Smartphones with Conditional Wasserstein Generative Adversarial Network," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 5447-5460, Apr. 2022.

[25] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," in *Proc. 31st Int. Conf. Neur. Info. Porcess. Sys. (NIPS)*, Dec. 2017, pp. 1-11.

[26] B.A. El-Rahiem, F.E.A. El-Samie, M. Amin, "Multimodal biometric authentication based on deep fusion of electrocardiogram (ECG) and finger vein," *Multimed. Syst.*, 2021, https://doi.org/10.1007/s00530-021-00810-9.

[27] Y. Li, P. Tao, S. Deng, G. Zhou, "DeFFusion: CNN-based Continuous Authentication Using Deep Feature Fusion," *ACM Trans. Sens. Netw.*, vol. 18, no. 2, Article 18, 20 pages, Oct. 2021.

[28] Y. Li, B. Zou, S. Deng and G. Zhou, "Using Feature Fusion Strategies in Continuous Authentication on Smartphones,"" *IEEE Internet Comput.*, vol. 24, no. 2, pp. 49-56, Mar.-Apr. 2020.

[29] Q. Zhang, H. Li, Z. Sun and T. Tan, "Deep Feature Fusion for Iris and Periocular Biometrics on Mobile Devices," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 11, pp. 2897-2912, Nov. 2018.

[30] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., "Generative adversarial nets," in *Proc. 27th Int. Conf. Neur. Info. Process. Sys. (NIPS)*, Dec. 2014, pp. 2672–2680.

[31] A. Radford, L. Metz, S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. 4th Int. Conf. Learning Representations (ICLR)*, May 2016, pp. 1-16.

[32] D. R. Lucio, R. Laroca, E. Severo, A. S. Britto Jr., D. Menotti, "Fully Convolutional Networks and Generative Adversarial Networks Applied to Sclera Segmentation," in *2018 IEEE 9th Int. Conf. Biometrics: Theory, Applications, and Systems (BTAS)*, Oct. 2018, pp. 1-7.

[33] T. R. Shaham, T. Dekel, T. Michaeli, "SinGAN: Learning a Generative Model from a Single Natural Image," in *IEEE/CVF Int. Conf. Computer Vision (ICCV)*, Oct. 2019, pp. 4570-4580.

[34] R. Ding, G. Guo, X. Yan, B. Chen, Z. Liu and X. He, "BiGAN: Collaborative Filtering with Bidirectional Generative Adversarial Networks," in *Proc. 2020 SIAM Int. Conf. Data Mining (SDM)*, Oct. 2020, pp. 82-90.

[35] T. Karras, T. Aila, S. Laine, J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," in *Proc. 6th Int. Conf. Learning Representations (ICLR)*, Apr. 2018, pp. 1-26.

[36] Y. Jiang, S. Chang, Z. Wang, "TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up," in *Proc. 35st Int. Conf. Neur. Info. Porcess. Sys. (NIPS)*, Dec. 2021, pp. 1–14.

[37] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, "Improved training of wasserstein GANs," in *Proc. 31st Int. Conf. Neur. Info. Porcess. Sys. (NIPS)*, Dec. 2017, pp. 5769–5779.

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *arXiv preprint arXiv:1704.04861*, 2017.

[39] X. Zhang, X. Zhou, M. Lin, J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6848-6856.

[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4510-4520.

[41] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, C. Xu, "ghostNet: More Features from Cheap Operations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1580-1589.

[42] C. Wu, K. He, J. Chen, Z. Zhao, R. Du, "Liveness is Not Enough: Enhancing Fingerprint Authentication with Behavioral Biometrics to Defeat Puppet Attacks," in *Proc. 29th USENIX Security Symposium (USENIX Security)*, Aug. 2020, pp. 2219-2236.

[43] Q. Yang *et al.*, "A multimodal data set for evaluating continuous authentication performance in smartphones," in *Proc. 12th ACM Conf. Embedded Netw. Sen. Syst. (SenSys)*, Nov. 2014, pp. 358–359.

[44] X. Chang, C. Peng, G. Xing, T. Hao and G. Zhou, "Isleep: A smartphone system for unobtrusive sleep quality monitoring," *ACM Trans. Sen. Netw.*, vol. 16, no. 3, Jul. 2020.

[45] Y. Li, H. Hu, Z. Zhu and G. Zhou, "SCANet: Sensor-based Continuous Authentication with Two-stream Convolutional Neural Networks," *ACM Trans. Sen. Netw.*, vol. 16, no. 3, Article 29, 27 pages, Jul. 2020.

[46] Y. Luo, B.-L. Lu, "EEG Data Augmentation for Emotion Recognition Using a Conditional Wasserstein GAN," in *2018 40th Ann. Int. Conf. IEEE Engineering in Medicine and Biology Society (EMBC)*, Jun. 2018, pp. 2535-2538.

[47] A. Gretton, K. M. Borgwardt, M. Rasch, B. Scholkopf, A. J. Smola, "A Kernel Method for the Two-Sample-Problem," in *Proc. 19th Int. Conf. Neural Info. Porcess. Sys. (NIPS)*, Dec. 2006, pp. 513–520.

[48] L. V. D. Maaten, G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[49] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM* vol. 60, no. 6, pp. 84–90, Jun. 2017.

[50] G. Huang, Z. Liu, L. V. D. Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proc. IEEE Conf. on Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2261-2269.

[51] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. on Comput. Vis. Pattern Recognit. (CVPR)*), 2015, pp. 1-9.

[52] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. on Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770-778.

[53] M. M. Breunig, H.-P. Kriegel, R. T. Ng and J. Sander, "LOF: identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, Jun. 2000.

[54] F. T. Liu, K. M. Ting and Z. Zhou, "Isolation Forest," *Proc. 8th IEEE Int. Conf. on Data Min.*, Dec. 2008, pp. 413-422.

[55] 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 Trans. Inf. Forensis Security*, vol. 8, no. 1, pp. 136–148, Jan. 2013.

[56] R. Dellana and K. Roy, "Data augmentation in CNN-based periocular authentication," in *Proc. 2016 6th Int. Conf. Inf. Com. Management (ICICM)*, Oct. 2016, pp. 141-145.

[57] G. Giorgi, A. Saracino and F. Martinelli, "Using recurrent neural networks for continuous authentication through gait analysis, " *Pattern Recognit. Lett.*, vol. 147, pp. 157-163, Jul. 2021.

[58] B. Zhou, J. Lohokare, R. Gao and F. Ye, "EchoPrint: Two-factor authentication using acoustics and vision on smartphones," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Oct. 2018, pp. 321–336.

[59] Y. Luo, L.-Z. Zhu, X.-Y. Wan, B.-L. Lu, "Data augmentation for enhancing EEG-based emotion recognition with deep generative models," *J. Neural Eng.*, vol. 17, no. 5, Article no. 056021.

[60] K. O. Bailey, J. S. Okolica, G. L. Peterson, "User identification and authentication using multi-modal behavioral biometrics," *Comput. Secur.*, vol. 43, pp. 77-89, 2014.

[61] I. Lamiche, G. Bin, Y. Jing, et al., "A continuous smartphone authentication method based on gait patterns and keystroke dynamics," *J. Ambient Intell. Human Comput.*, vol. 10, pp. 4417–4430, 2019.