SCANet: Sensor-based Continuous Authentication with Two-stream Convolutional Neural Networks

YANTAO LI, College of Computer Science, Chongqing University, China
HAILONG HU and ZHANGQIAN ZHU, Southwest University, China
GANG ZHOU, Department of Computer Science, William & Mary, USA

Continuous authentication monitors the security of a system throughout the login session on mobile devices. In this paper, we present SCANet, a two-stream convolutional neural network based continuous authentication system that leverages the accelerometer and gyroscope on smartphones to monitor users' behavioral patterns. We are among the first to use two streams of data - frequency domain data and temporal difference domain data - from the two sensors as the inputs of the convolutional neural network (CNN). SCANet utilizes the two-stream CNN to learn and extract representative features, and then performs the principal component analysis (PCA) to select the top 25 features with high discriminability. With the CNN-extracted features, SCANet exploits the one-class support vector machine (one-class SVM) to train the classifier in the enrollment phase. Based on the trained CNN and classifier, SCANet identifies the current user as a legitimate user or an impostor in the continuous authentication phase. We evaluate the effectiveness of the two-stream CNN and the performance of SCANet on our dataset and BrainRun dataset, and the experimental results demonstrate that CNN achieves 90.04% accuracy, and SCANet reaches an average of 5.14% equal error rate (EER) on two datasets and takes approximately 3 seconds for user authentication.

CCS Concepts: • **Security and privacy** → **Authentication**; *Mobile platform security*; • **Human-centered computing** → Collaborative interaction; • **Computing methodologies** → Neural networks.

Additional Key Words and Phrases: Continuous authentication, accelerometer and gyroscope, two-stream convolutional neural network (CNN), one-class support vector machine (SVM), equal error rate (EER)

ACM Reference Format:
Yantao Li, Hailong Hu, Zhangqian Zhu, and Gang Zhou. 2020. SCANet: Sensor-based Continuous Authentication with Two-stream Convolutional Neural Networks. *ACM Trans. Sensor Netw.* 1, 1, Article 1 (April 2020), 27 pages. https://doi.org/10.1145/3397179

1 INTRODUCTION

With the widespread usage of mobile devices, a variety of mobile users use their devices to store private and sensitive information, share personal information, or conduct commercial transactions. To prevent users' critical information on mobile devices from leaking or being illegally accessed, user authentication mechanisms have been developed and applied. Authentication refers to the process of verifying a user based on certain credentials before granting access to a secure system or resource [1, 2]. Typical one-time user authentication mechanisms, such as passwords (e.g. PINs), graphical patterns, touch ID, and even face ID, have been widely deployed on mobile/smart devices. However,

Authors' addresses: Yantao Li, College of Computer Science, Chongqing University, 174 Shapingba Central St, Chongqing, 400044, China, yantaoli@cqu.edu.cn; Hailong Hu; Zhangqian Zhu, Southwest University, 2 Tiansheng Rd, Chongqing, 400715, China; Gang Zhou, Department of Computer Science, William & Mary, 251 Jamestown Rd, Williamsburg, VA, 23185, USA.

1550-4859/2020/4-ART1 $15.00
https://doi.org/10.1145/3397179

these mechanisms authenticate users only at the time of initial logging-in, and serious security flaws arise after the initial authentication has been performed. For instance, an unauthorized user can easily gain access to an unattended mobile device without logging out [3]. These security flaws have led to the investigation of continuous authentication mechanisms.

Continuous authentication has been a promising mechanism to alleviate the above security flaws, by frequently authenticating users via biometrics-based approaches. These approaches can be broadly categorized into: physiological biometrics-based approaches and behavioral biometrics-based approaches. Specifically, on the one hand, the physiological biometrics-based approaches rely on static physical attributes, such as iris patterns [4], fingerprints [5], pulse [6], voice [7, 8] and face patterns [9, 10], but these approaches require direct user participation in the process of the authentication. On the other hand, behavioral biometrics-based approaches exploit user behavioral patterns, such as touch gestures [11, 12], gait [13, 14], and GPS patterns [15, 16]. In addition, deep learning technologies have been applied to smart devices for user continuous authentication [17–19]. These approaches identify invariant features of user interactions with smart devices by using sampling data from built-in sensors and accessories, such as the accelerometer, gyroscope, magnetometer, and touch screen.

However, current continuous authentication mechanisms on smart devices are primarily facing two challenges: feature robustness and system effectiveness. On the one hand, it is hard to capture the most robust features that accommodate diverse noise patterns since sensor data collected by smartphones contain much noise [20, 21]. For instance, the authors in [22, 23] utilize designed features, such as touch-screen features and HMOG features, to achieve low authentication accuracy with 12.85% and 7.16% EERs, respectively. On the other hand, it is not easy to design an effective continuous system with less limitations, such as representational power of extracted features, and computational cost [4–6, 24]. For example, the authors of [25, 26] exploit the trained Hidden Markov Model and one-class SVM classifiers to conduct the authentication within approximately 8s and 5s, respectively, based on the designed statistical features. To address the first challenge, we generate two-stream data from the built-in sensors: frequency domain data and temporal difference domain data. The frequency domain data are converted from time domain data by applying Fourier transform, which contain better local frequency patterns that not only alleviate the impact of noise but are also independent of how time-series data are organized in the time domain [27]. The temporal domain data are generated by calculating the difference of the time domain data in two consecutive time intervals, which contain dynamic temporal features [28]. Then, based on the sensor data, we design a two-stream convolutional neural network (two-stream CNN) to learn and extract representative features with high discriminability to achieve an average of 5.14% EER on our dataset and BrainRun dataset. For the other challenge, we utilize temporal difference and frequency data to generate CNN-extracted features and then exploit the trained one-class SVM classifier to authenticate users within 3s. In addition, the two-stream CNN adapts to resource-constrained mobile devices by significantly decreasing network parameters and the number of operations while maintaining the same accuracy.

In this paper, by extending our previous work [29], we present SCANet, a novel continuous authentication system based on a two-stream convolutional neural network (two-stream CNN) that leverages the accelerometer and gyroscope on smartphones to monitor users' behavioral patterns. Specifically, SCANet consists of five modules: data collection, data preprocessing, feature extraction, classification, and authentication. The operation of SCANet includes the enrollment phase (for data collection, feature extraction by the two-stream CNN, and classifier training), and the continuous authentication phase (for classifier testing and authentication). In the enrollment phase, the data collection module captures users' behavioral patterns during smartphone usage, by utilizing the two sensors of the accelerometer and gyroscope that are omnipresently built into smartphones.

The data preprocessing module converts the collected time domain data into two streams of data - frequency domain data and temporal difference data - that are used as the inputs of the CNN. In the feature extraction module, we design a two-stream CNN to learn and extract the representative features for resource-constrained mobile devices and then apply the principal component analysis (PCA) to these features in order to select the top 25 features with high discriminability. With the CNN-extracted features, we use the one-class support vector machine (one-class SVM) to train the classifier. In the continuous authentication phase, based on the trained CNN and classifier, SCANet classifies the current user as a legitimate user or an impostor. Note that the two-stream CNN is only trained based on a legitimate user's data in the enrollment phase, and then the trained CNN is used as a feature extractor in the continuous authentication phase. We evaluate the effectiveness of the two-stream CNN in terms of five metrics (accuracy, macro F1, micro F1, model parameters, and computational cost), and evaluate the performance of SCANet with respect to four metrics (equal error rate, false acceptance rate, false rejection rate, and time efficiency) and unseen users on our dataset, respectively, and the accuracy performance on BrainRun dataset. The experimental results indicate that the two-stream CNN achieves 90.04% accuracy, 85.05% macro F1, and 90.04% micro F1 in the 5-second time window, and 1.8M model parameters and 120M computational cost in the 2-second time window. SCANet reaches an average of 4.57% equal error rate (EER), 4.65% false acceptance rate (FAR) and 4.48% false rejection rate (FRR) on our dataset, and an average of 5.71% EER, 5.87% FAR and 5.56% FRR on BrainRun dataset. SCANet takes approximately 3 seconds for user authentication.

The main contributions of this work are summarized as follows:

- We design SCANet, a two-stream convolutional neural network based continuous authentication system that authenticates smartphone users by leveraging the built-in accelerometer and gyroscope to monitor users' behavior patterns. SCANet is composed of five modules: data collection, data preprocessing, feature extraction, classification, and authentication.
- We propose a two-stream CNN based on the depthwise separable convolution and linear bottlenecks, which uses frequency domain data and temporal difference data collected by the accelerometer and gyroscope as its two-stream inputs to learn and extract representative features with high discriminability.
- We evaluate the effectiveness of the two-stream CNN and the performance of SCANet on our dataset and BrainRun dataset, and the experimental results demonstrate that the two-stream CNN achieves an accuracy of 90.04%, and SCANet reaches an average of 5.14% EER on the two datasets and takes approximately 3 seconds for user authentication.

The remainder of this paper is organized as follows: Sec. 2 reviews the-state-of-art of efficient network architectures and continuous authentication systems, and Sec. 3 proposes the two-stream CNN to learn and extract representative features with high discriminability. In Sec. 4, we detail the architecture of SCANet in data collection, data preprocessing, feature extraction, classification, and authentication. In Sec. 5, we describe the dataset, classifier training, and accuracy metrics for experiments, and evaluate the performance of the two-stream CNN and SCANet in Sec. 6. Finally, we conclude the work in Sec. 7.

2 RELATED WORK

In this section, we review the state-of-the-art of efficient network architectures and continuous authentication systems.

2.1 Efficient Network Architecture

Deep neural networks have become one of the most popular methodologies in the area of the artificial intelligence, such as speech recognition [30] and computer vision [31], in recent years. There are some works devoting to detecting concrete crack damage in images by utilizing deep learning technology, such as *Faster R-CNN* [32] and *CNN-CDD* [33]. For instance, to provide quasi real-time simultaneous detection of multiple types of damages, *Faster R-CNN* proposes a faster region-based convolutional neural network-based structural visual inspection method. A large number of efficient network architectures have been proposed to improve the accuracy, such as *ResNet* [31], *AlexNet* [34], and *VGGNet* [35]. For example, *ResNet* proposes shortcut connections for CNNs, which greatly reduces the difficulty of training super-deep models. However, since *ResNet* mainly focuses on visual inputs and super-deep models, it is not suitable for sensor data input and can not be performed on the computationally limited platforms, such as smartphones. There are some works dedicated to tuning neural network architectures to reach an optimal trade-off between the accuracy and performance on some mobile and embedded applications, such as *MobileNet* [36], *ShuffleNet* [37] and *MobileNetV2* [38]. For instance, *MobileNetV2* is based on an inverted residual structure and achieves the state-of-the-art performance in COCO object detection, ImageNet classification, VOC image segmentation. However, it also ignores sensor data inputs.

Our work differs in that the specially designed two-stream CNN architecture mainly deals with multi-sensor inputs, and can be performed on the computationally limited platform, such as smartphones.

2.2 Continuous Authentication System

Most authentication mechanisms on smartphones, such as passcodes, graphical patterns, Touch IDs, and even face IDs, provide security just by a one-time authentication, which enable unauthorized users to easily gain access to unattended mobile devices without logging out. To alleviate this security issue, continuous authentication mechanisms are explored and developed, which can be broadly categorized into two groups: physiological biometrics-based approaches and behavioral biometrics-based approaches. Specifically, on the one hand, the physiological biometrics-based approaches authenticate users by static physical attributes, such as iris patterns [4], fingerprints [5], pulse [6], voice [7, 8], and face patterns [9, 10]. In [4], the authors provide two different iris recognition approaches using Gabor filters and multiscale zero-crossing representation. The authors in [5] develop a prototype biometrics system that integrates faces and fingerprints and operates in the identification mode with an admissible response time. In [6], the authors propose a biometric based on the human body's response to an electric square pulse signal that can be used to enhance security in an additional authentication mechanism in PIN entry systems, and a means of continuous authentication on a secure terminal. The authors in [8] propose a continuous authentication system *VAuth* through executing only the commands that originate from the voice of the owner. In [10], the authors propose an application of the scale invariant feature transform approach in the context of the face authentication. However, these approaches require users' direct participation in the process of the authentication. On the other hand, the behavioral biometrics-based approaches authenticate users by the invariant features of human behaviors during different activities, such as touch gestures [11, 12], gait [13, 14] and GPS patterns [15, 16]. In [12], the authors propose a touch-based authentication system by exploiting a novel one-class classification algorithm import vector domain description during smartphone usage. The authors in [13] identify users of portable devices from gait pattern with accelerometers by using the acceleration signal characteristics produced by walking. In [15], the authors present a n-gram based model for modeling a user's mobility patterns. However, these approaches can not achieve better performance due to lacking of

robust features or efficient algorithms for authentication. In addition, deep learning technologies have been applied to smart devices for user continuous authentication [17–19]. The authors in [17] propose a Siamese convolutional neural network-based continuous authentication system that detects unauthorized use on the smartphone by using the characteristic motion patterns of each individual interacting with the device. In [18], the authors present a deep learning autoencoder-based continuous biometric authentication system that relies on user-specific motion patterns while interacting with the smartphone. The authors in [19] propose a deep recurrent neural network-based authentication framework that leverages the unique motion patterns when users entering passwords as behavioural biometrics.

We are different in that we design a two-stream CNN adaptable for mobile platforms to learn and extract representative features that achieve better performance for continuous user authentication.

3 TWO-STREAM CNN ARCHITECTURE

Learning representative features is critical to continuous authentication systems because the performance of real-word systems conforms to a set of criteria, such as representational power of extracted features, and speed of the feature extractor. In this section, we introduce the depthwise separable convolutions, and detail the structure of linear bottlenecks. Then, we elaborate on the architecture of the two-stream CNN and describe its ability to learn representative features with less network parameters and less operations while maintaining the authentication accuracy [39].

3.1 Depthwise Separable Convolution

Depthwise separable convolutions (DSC) factorize a standard convolution into a depthwise convolution and a pointwise (or 1×1) convolution. The two-stream CNN model is based on depthwise separable convolutions, where the depthwise convolution applies a single convolutional filter to each input channel, and where the pointwise convolution applies a 1×1 convolution to build new features (through computing linear combinations of the input channels).

A standard convolution takes a $c_i \times h_i \times w_i$ input tensor $\mathbf{L_i}$ (where c_i is the number of channel i, and h_i and w_i are the height and the width) and applies a convolutional kernel $K \in R^{c_j \times c_i \times k_1 \times k_2}$ to produce a $c_j \times h_i \times w_i$ input tensor $\mathbf{L_j}$. The computational cost of a standard convolutional layer is $h_i \times w_i \times c_i \times c_j \times k_1 \times k_2$. Although depthwise separable convolutions empirically work as well as standard convolutions, they only cost $\frac{k_1 \times k_2 \times c_j}{k_1 \times k_2 + c_j}$, which is the sum of the depthwise and pointwise convolutions. Compared with a standard convolutional layer, the two layers of the depthwise separable convolution reduce computation by almost a factor of $k_1 \times k_2$, where k_1 and k_2 are the sizes of convolutional kernel. In this work, SCANet uses 1×32 depthwise separable convolution for convolutional layers; therefore, the computational cost is 32 times smaller than that of standard convolutions with only a small reduction in accuracy [26, 42, 43].

3.2 Linear Bottleneck

Our linear bottlenecks are based on the depthwise separable convolutions as described above. The structure of these linear bottlenecks is shown in Fig. 1 and Table 1. A block with size $h_i \times w_i$, kernel size $k_1 \times k_2$, expansion factor t, c_i input channels, and c_j output channels takes a low-dimensional input $c_i \times h_i \times w_i$. The block is first expanded to a high dimension by the expansion factor t and then filtered with a lightweight depthwise separable convolution with kernel size $k_1 \times k_2$ and stride 1, both of which use Rectified Linear Unit (ReLU6) as their activation function. Then the block's features are projected back to a low-dimensional output $c_j \times h_i \times w_i$ with a linear convolution.

We construct the two-stream CNN based on linear bottleneck blocks, because: 1) they greatly reduce the number of parameters and computational cost in convolutional operations; 2) they

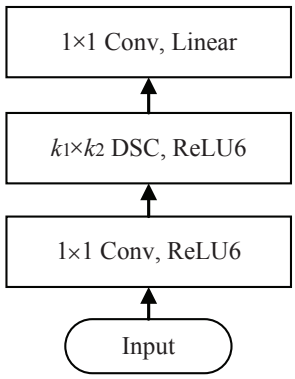

Fig. 1. Structure of Linear Bottlenecks.

Table 1. Bottleneck block transforming from c_i to c_j channels, with stride s, and expansion factor t.

Input	Operator	Output
$c_i \times h_i \times w_i$	1×1 Conv, ReLU6	$tc_i \times h_i \times w_i$
$tc_i \times h_i \times w_i$	1×3 DSC, s=1, ReLU6	$tc_i \times h_i \times w_i$
$tc_i \times h_i \times w_i$	1×1 Conv, Linear	$c_j \times h_i \times w_i$

provide a natural separation between the input/output domains of the bottleneck layers and the layer transformation [38]. In this work, we do not use skip connection since we set $stride = 1$. The total number of multiplications and additions required is $h_i \times w_i \times c_i \times t \times \left(c_i + k_1 \times k_2 + c_j\right)$.

3.3 Two-stream CNN

Based on the linear bottlenecks, we construct the two-stream CNN architecture as illustrated in Fig. 2 and detailed in Table 2. The two-stream inputs of CNN are the frequency domain data $\mathbf{X_f}$ and the temporal difference data $\mathbf{X_t}$, which are elaborated in data prepressing (Sec. 4.2). As shown in Fig. 2, the CNN structure consists of two individual convolutional subnets (subnet 1 and subnet 2) and a single merged convolutional subnet. With the inputs of the frequency domain data and temporal difference data, the two-stream CNN extracts three kinds of features/relationships embedded in $\mathbf{X_f}$ and $\mathbf{X_t}$: the features in the frequency domain, the features in the temporal domain, and the relationships across sensor data dimensions. The frequency domain generally includes many spatial patterns in some neighboring frequencies. The temporal domain commonly contains a number of local temporal dynamic patterns. The interaction among sensor data usually contains all dimensions.

As demonstrated in Fig. 2, for $\mathbf{X_f}$ (Individual convolutional subnet 1), we first apply $2d$ filters with shape $(d, Conv1)$ to learn interaction among sensor data dimensions and spatial patterns in the frequency domain. Then, we apply two linear bottlenecks with shapes $(1, Conv2)$ and $(1, Conv3)$ hierarchically to learn high-level features. For $\mathbf{X_t}$ (Individual convolutional subnet 2), the same process applies, since the structures of the individual convolutional subnets are the same. Next, we flatten the two outputs and concatenate them along channels. For the merged convolutional subnet, we first apply $2d$ filters with shape $(2, Conv4)$ to learn the interactions between the temporal and frequency domains, and then hierarchically apply Linear Bottlenecks with shapes $(1, Conv5)$ and $(1, Conv6)$ to learn high-level features. Finally, we use two full connection layers (layer 1 and layer

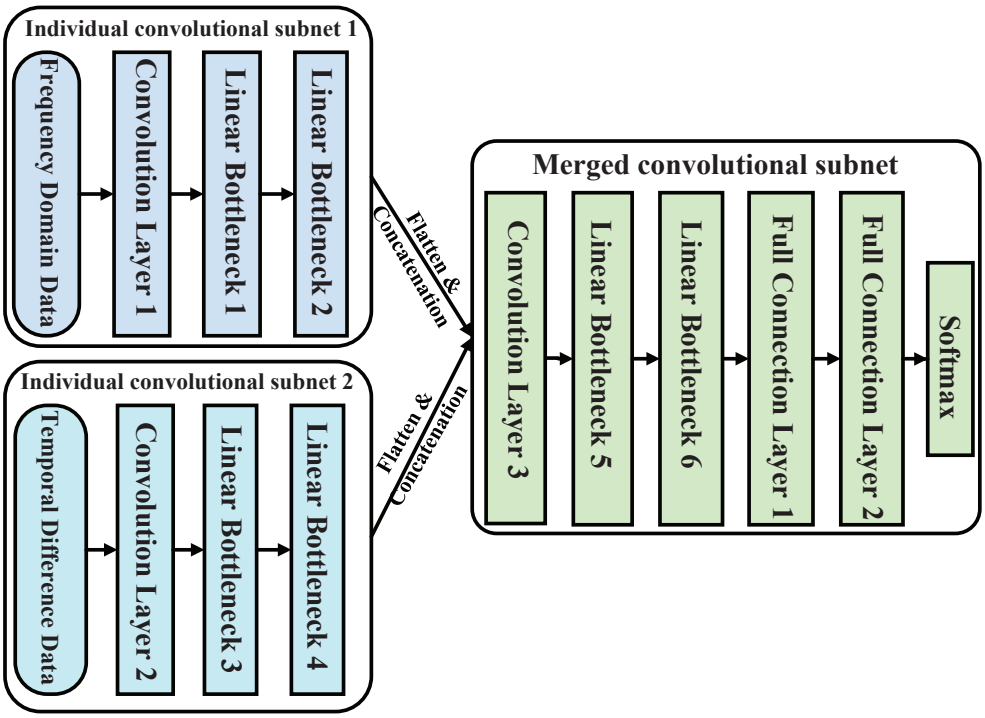

Fig. 2. Architecture of the two-stream CNN.

2) to classify the input patterns into a finite number of classes. The output of these full connection layers is ultimately fed into a softmax layer to generate the predicted category probability.

As detailed in Table 2, two-stream CNN learns 32 filters for individual convolutional subnets, and 64 filters for the merged convolutional subnet. For all the experiments, expansion factor $t = 6$. That is, for a bottleneck layer, when the input and output tensors have 32 channels and 64 channels, respectively, then the intermediate expansion layer has 192 (32×6) channels. We use ReLU6 as our activation function due to its robustness even in low-precision computation [36]. In addition, batch normalization is applied at each layer to reduce internal covariate shift. Dropout is employed during training process. Note that the structures of the individual convolutional subnets for two streams of inputs are the same, but their parameters are not shared and they are learned separately, and the symbol "-" in the table indicates there is no corresponding parameter value.

Also, note that a sequence of sensor data can be represented as various formats. In our experiment, it is represented as a tensor with shape $T \times w \times h$. If the number of the sensor is fixed, w and h are constants. However, the number of time window T can be regarded as a tunable hyper parameter, which can be adjusted depending on desired accuracy/performance trade-offs. Our primary network ($8 \times 75 \times 6$), has a computational cost of 120 million multiplication-addition operations (MAdds) and uses 1.8 million parameters. We also explore the performance trade-offs, for the number of time window from 8 (2 seconds) to 20 (5 seconds). The network computational cost ranges from $120M$ MAdds to $310M$ MAdds, while the model size varies between $1.8M$ and $4.16M$ parameters. For convenience, one minor implementation difference is the input tensor with shape $1 \times T \times (2f \cdot d)$ and $1 \times T \times (\tau \cdot d)$ to avoid $3d$ convolutional kernel.

Table 2. The two-stream CNN body architecture

Network	Input	Operator	t	# Kernel	Stride	Padding	Kernel
Individual convolutional subnet 1	$1 \times 8 \times 300$	Conv2d	-	32	(1, 3)	(0, 0)	(1, 3× 3)
	$32 \times 8 \times 98$	Bottleneck	6	32	(1, 3)	(0, 0)	(1, 3× 2)
	$32 \times 8 \times 31$	Bottleneck	6	32	(1, 3)	(0, 0)	(1, 3× 1)
Individual convolutional subnet 2	$1 \times 8 \times 150$	Conv2d	-	32	(1, 3)	(0, 0)	(1, 3× 3)
	$32 \times 8 \times 48$	Bottleneck	6	32	(1, 3)	(0, 0)	(1, 3× 2)
	$32 \times 8 \times 15$	Bottleneck	6	32	(1, 3)	(0, 0)	(1, 3× 1)
Flatten and concatenate the outputs of the subnets 1 and 2							
Merged convolutional subnet	$1 \times 8 \times 480$	Conv2d	-	64	(1, 2)	(0, 1)	(1, 16× 2)
	$64 \times 8 \times 226$	Bottleneck	6	64	(1, 2)	(0, 1)	(1, 16× 2)
	$64 \times 8 \times 99$	Bottleneck	6	64	(1, 2)	(0, 1)	(1, 16× 1)
	$64 \times 8 \times 43$	Avgpool (1, 7)	-	-	-	-	-
	1×3072	Full connection	-	-	-	-	-
	1×512	Full connection	-	-	-	-	-

4 SCANET DESIGN

In this section, we present the design of the sensor-based continuous authentication system using the two-stream convolutional neural network, SCANet, to continuously monitor users' behavioral patterns by leveraging the accelerometer and gyroscope on smartphones. We illustrate the architecture of SCANet in Fig. 3. As demonstrated in Fig. 3, SCANet consists of five modules: data collection, data preprocessing, feature extraction, classification, and authentication. The operation of SCANet includes two phases for learning and classifying users' behavioral patterns: the enrollment phase and the continuous authentication phase, where SCANet learns the profile of a legitimate user in the enrollment phase and then authenticates users in the continuous authentication phase. We describe the five modules in the following:

4.1 Data collection

The data collection module collects all users' sensor data from the accelerometer and gyroscope. The accelerometer records a user's motion patterns, such as arm movements or gaits, and the gyroscope records a user's fine-grained motions, such as smartphone orientation during usage. The two sensors do not require root permission when requested by mobile applications, which makes them useful in background monitoring. In *SCANet*, the data collection module captures the user's every subtle movement during operation on their smartphones, and records the instantaneous readings of the two sensors when the screen is on. The collected data are stored in a protected buffer for data preprocessing.

4.2 Data preprocessing

The synchronized raw sensor readings of the accelerometer and gyroscope at a time point can be represented by a vector $\mathbf{w}= \left(x_{acc}, y_{acc}, z_{acc}, x_{gyro}, y_{gyro}, z_{gyro}\right)^T \in R^6$, where x, y and z represent the three-axis sensor readings of a sensor, and acc and $gyro$ indicate the accelerometer and gyroscope, respectively. For a series of sensor readings over certain time, they can be represented by a $d \times N$ matrix $\mathbf{P}= (\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_N})$, where d is the dimension of sensor readings ($d = 6$) and N is the number of raw sensor readings over the time.

We first segment the sensor readings $\mathbf{P}$ into a series of non-overlapping time intervals with width τ (τ represents the number of sensor data in a time interval). In each time interval, a matrix

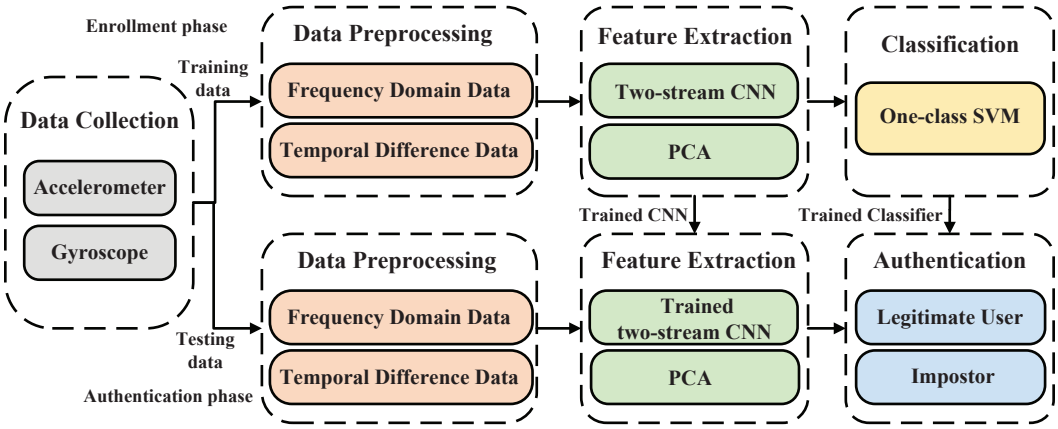

Fig. 3. Architecture of SCANet.

$\mathbf{Q} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_\tau)$ has a shape $d \times \tau$. For frequency domain data input, we apply Fourier transform to each element in $\mathbf{Q}$ in a time interval, and then stack these outputs into a $d \times 2f$ matrix $\mathbf{X_f}$, where f is the dimension of frequency domain containing f magnitude and phase pairs ($f = \tau$). For temporal difference data input, we compute the difference of the time domain data in two consecutive time intervals and obtain a $d \times \tau$ matrix $\mathbf{X_t}$. For example, we formulate the readings in time interval T as $\mathbf{X_t^T} = \mathbf{Q_{T+1}} - \mathbf{Q_T}$. Finally, $\mathbf{X_f}$ and $\mathbf{X_t}$ have shapes of $n \times T \times d \times 2f$ and $n \times T \times d \times \tau$, respectively, where $n = N/(T \times \tau)$ is the number of samples fed into the efficient CNN, and T is the number of time intervals for a time window. The length of a time window T usually determines the time that the system requires to perform the continuous authentication.

After data preprocessing, the frequency domain data $\mathbf{X_f}$ and temporal difference data $\mathbf{X_t}$ are obtained as the two-stream inputs for the CNN.

4.3 Feature Extraction

With the two-stream inputs of the frequency domain data $\mathbf{X_f}$ and temporal difference data $\mathbf{X_t}$, the feature extraction module extracts features with high discriminability through feature learning and feature selection. The feature extraction module first learns representative features by using the two-stream CNN and then selects highly discriminative features from the these representations via the principal component analysis (PCA).

- **Feature learning**: The two-stream CNN concentrates on learning representative features by detecting spatial patterns (related to the frequency domain data) and local temporal dynamic patterns (based on the time difference domain data).
- **Feature selection**: Based on the extracted features, the top 25 features with high discriminability are selected using the PCA.

4.4 Classification

The CNN-extracted features are then fed to the classifier for training and classifying. We use the one-class support vector machine (one-class SVM) classifier, which utilizes a kernel function to map data into a high dimensional space, and which considers the origin as the only sample from other classes [44]. In the enrollment phase, the classifier is established by using training feature vectors with a radial basis function (RBF) kernel. In the continuous authentication phase, the trained

classifier projects the testing feature vectors onto the same high-dimensional space and classifies the testing feature vectors.

4.5 Authentication

Based on the testing features from trained two-stream CNN and the trained one-class SVM classifier, the authentication module classifies the current user as a legitimate user or an impostor. In the enrollment phase, the legitimate user's profile is generated from the training data and stored in a protected buffer, while the current user's features are compared with the profile in the authentication phase. If the current user is classified as an impostor, *SCANet* will require initial login inputs; otherwise, it will continuously authenticate the user.

5 EXPERIMENT

In this section, we first detail how to collect the dataset, then explain how to train the one-class SVM classifier, and finally elaborate the accuracy metrics for performance evaluation.

5.1 Dataset

To investigate the authentication accuracy of SCANet, we developed a data collection tool for Android phones to record real-time behavioral data invoked by user's interaction with the phone. We recruited 100 volunteers (53 male, and 47 female) to conduct three tasks on the phones, including document reading, text production, and navigation on a map to locate a destination. When logging into the data collection tool, the volunteer is randomly assigned a reading, writing, or map navigation session. One session lasts 5 to 15 minutes, and each volunteer is expected to perform 24 sessions (8 reading sessions, 8 writing sessions, and 8 map navigation sessions). In total, each volunteer contributes 2 to 6 hours of behavior traits. The collected data are stored in CSV files on the phone [23]. We recorded sensor readings of the accelerometer and gyroscope with the sampling rate of $100Hz$ and selected the first 100 minutes of the data for each user with 2-second and 5-second window sizes as the experiment dataset.

5.2 Training

The training phase of *SCANet* is divided into two stages. In the first stage, the two-stream CNN is trained to learn the universal features and relationships among the sensors for classification. We use a step decay strategy to anneal the learning rate over time. The learning rate is initially set to 0.0001, and then is gradually reduced by a 0.95 learning rate decay factor when the accuracy starts to decline. A ReLU6 is taken as the activation function. The batch size is set to 256, and the network is trained for up to 100 epochs. We stop the epoch when loss function does not decrease for ten consecutive training epochs.

The second stage is devoted to training *SCANet*. The trained two-stream CNN is fixed as a universal feature extractor, and we train the one-class SVM using ten-fold cross validation. We specify one of the 100 users as a legitimate user and the rest as impostors. In so doing, we ensure the positive feature samples from one legitimate user and the negative feature samples from 99 impostors. Based on these samples, we train the classifier as follows:

Step 1: We randomly divide all positive samples into k ($k = 10$) equal-size subsets, where $k - 1$ positive subsets are used to train the one-class SVM model, and one subnet to test the model.

Step 2: We randomly select negative samples with the same size to positive ones from all the negative samples, which are also divided into k ($k = 10$) equal-size subsets. One of the 10 negative subsets is exploited to test the model.

Step 3: The above 2 steps are repeated 10 times until each subset of negative samples and each subsets of positive samples are tested exactly once.

Step 4: We repeat steps 1, 2, and 3 twenty times to account for randomness.

5.3 Accuracy Metrics

We describe six metrics that are used for analyzing the authentication accuracy of SCANet: accuracy, Micro F1 score, Macro F1 score, false acceptance rate (FAR), false rejection rate (FRR), and equal error rate (EER).

- **Accuracy**: It indicates the number of true positive (TP) divided by the number of all the authentication attempts.
- **Micro F1 score and macro F1 score**: F-measure accuracy (F1 score) is known as the harmonic mean of precision and recall. Precision is the number of true positive (TP) divided by the number of positive calls (TP+FP), while recall is the number of TP divided by the number of condition positives (TP+FN). Here, FP indicates false positive and FN represents false negative. F1 score ranges from [0, 1], where 0 indicates the worst and 1 the best. F1 score is defined as: $F_1(\%) = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2TP}{2TP+FP+FN}$. When dealing with multiple classes, there are two possible ways of averaging precision, recall, and F1-measure: micro F1 score and macro F1 score. Micro F1 score calculates the F1 score globally by counting the total true positives, false negatives, and false positives. This equally weights all the classes, thus favouring the performance on common classes. Macro F1 score calculates the F1 score for each class and finds their unweighted mean, which equally weights all the classes, regardless of how many documents belonging to it.
- **FAR, FRR and EER**: False acceptance rate (FAR) indicates the ratio of the number of false acceptances to the number of authentication attempts by impostors, while false rejection rate (FRR) represents the ratio of the number of false rejections to the number of authentication attempts by legitimate users [45]. Then, equal error rate (EER) is the point where FAR equals to FRR.

6 EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of the CNN and the performance of SCANet, respectively. We have implemented our system SCANet on smartphones by utilizing Python. The two-stream CNN and some representative algorithms, such as KRR, one-class SVM, and *k*NN, mainly call the Pytorch framework and sklearn package. The two-stream CNN in the enrollment phase is implemented on GPU using Inter Xeon E5-2683v3 server clocked at 2GHz with 512GB RAM and a NVIDIA Tesla M40 GPU with 12GB GDDR5 memory and 3072 CUDA cores. However, all the experiments are conducted on a single CPU clocked at 2GHz platforms, which is relatively low standard comparison with the CPU configuration of current smartphones. Before conducting the evaluation, we first elaborate the representative algorithms or models for comparison.

6.1 Algorithms for comparison

In this section, we first describe typical models for comparison with the two-stream CNN, and then introduce representative algorithms for comparison with SCANet.

6.1.1 Comparison models for the two-stream CNN. We compare our two-stream CNN model with other typical models. There are four variants of the CNN model according to input streams: Time-CNN, Frequency-CNN, TD-CNN, and RawFrequency-CNN. For convenient comparison, we refer to our two-stream CNN as TDFrequency-CNN in this section. In addition, we select two competitive models DeepSense and IDNet for comparison.

- **Time-CNN**: Time domain sensor data are used as a single-stream input for CNN. This model includes only one individual convolutional subnet [46].
- **Frequency-CNN**: Frequency domain sensor data are used as a single-stream input for CNN [47]. This model includes one individual convolutional subnet.
- **TD-CNN**: Temporal difference data are used as a single-stream input for CNN. This model includes one individual convolutional subnet.
- **RawFrequency-CNN**: Raw time domain data and frequency domain data are used as two-stream inputs of CNN. This model consists of two individual convolutional subnets and one merged convolutional subnet [29].
- **TDFrequency-CNN**: Temporal difference data and frequency domain data are used as two-stream inputs of CNN (details in Sec. 3.3)
- **DeepSense**: Frequency representations of sensor data are used as a single-stream input for the model. This model integrates convolutional neural networks and recurrent neural networks [48].
- **IDNet**: Time domain sensor data are used as a single-stream input of the model. This model utilizes convolutional neural networks including two convolutional layers and two fully connected layers [49].

6.1.2 Comparison algorithms for SCANet. We compare SCANet with other representative authentication algorithms. These authentication algorithms are: kernel ridge regression (KRR), one-class support vector machine (one-class SVM), support vector machine (SVM), and k-nearest neighbors (kNN), respectively. In addition, we also detail how to train these competitive algorithms in the enrollment phase and how to test them in the continuous authentication phase.

- **KRR**: The kernel ridge regression classifier is the combination ridge regression (linear least squares with 12-norm regularization) and the kernel trick. It learns a linear function in the space induced by the kernel function and data. In the enrollment phase, the classifier is trained by the training vectors with the RBF kernel function, which is similar to a SVM. KRR parameters and kernel parameters are set by grid search [16]. In the continuous authentication phase, the classifier maps the testing vector by the RBF kernel function into the high-dimension space to calculate the distance between the testing vector and the linear separator as the classification score.
- **One-class SVM**: The one-class SVM classifier is regarded as an unsupervised learning algorithm. Different from SVM, KRR, and kNN, it projects data onto a high dimensional space through a kernel function, and regards the origin as the only sample from other classes [50]. In the enrollment phase, the classifier is trained by the training vectors with the RBF kernel function, and one-class SVM parameters and kernel parameters are set by grid search. In the continuous authentication phase, the classifier maps the testing vector into the same high-dimension space. It calculates the distance between the testing vector and the linear separator as the classification score. Note that the difference between one-class SVM algorithm and SCANet is that SCANet uses the two-stream CNN to learn features, and then exploits the one-class SVM as the classifier.
- **SVM**: The support vector machine classifier is similar to the one-class SVM, but it is a classification algorithm. We consider it as a binary classifier in this work. In the enrollment phase, the classifier is trained by the training vectors from a legitimate user and impostors with the RBF kernel function, and the SVM parameter and kernel parameter are set by grid search. In the continuous authentication phase, the classifier maps the test vector into the same high-dimension space. This calculates the distance between the test vector and the linear separator as the classification score.

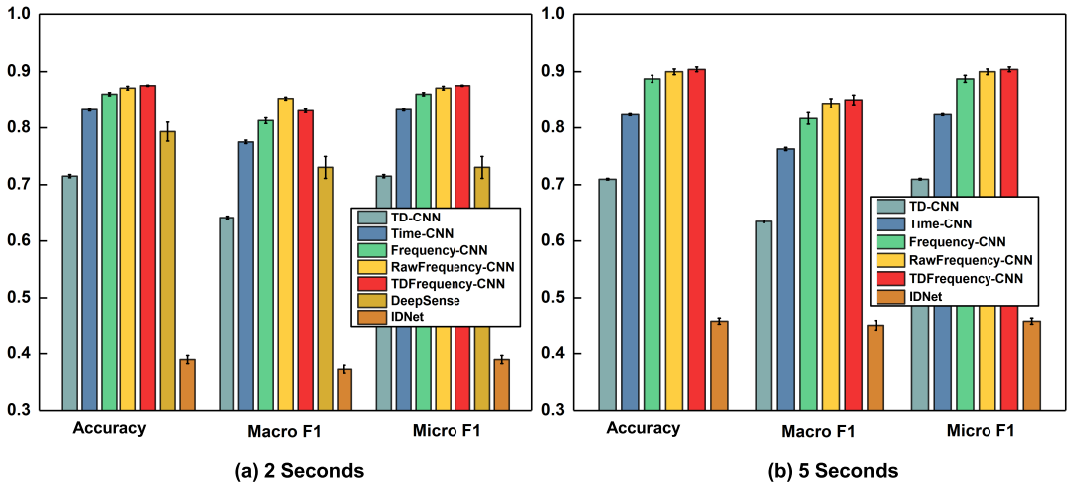

Fig. 4. Accuracy comparison results

- *k*NN: The classifier authenticates a user through an assumption that the testing vector from user will resemble one or more of those in the training vectors [51]. In the enrollment phase, the classifier estimates the covariance matrix of training vectors, and the nearest-neighbor parameter k is set by grid search. In the continuous authentication phase, the classifier computes Mahalanobis distance and the average distance from the testing vector to the nearest samples, which is used as the classification score.

6.2 Effectiveness of the two-stream CNN

In this section, to validate the effectiveness of the two-stream CNN, we evaluate its performance by comparing other competitive models with respect to accuracy, computational cost, and model size. For the comparison, we train the competitive models by the same process as ours.

6.2.1 Accuracy, macro F1 and micro F1. The two-stream CNN is considered as a multi-class classification problem (100 classes) and trained by Adam algorithm that minimizes a categorical cross-entropy loss function L, which is defined as $L = H(y, F(\chi))$, where $H(x, y)$ is the cross entropy for two distributions. All the results (accuracy, macro F1 and micro F1 scores) are average values of 5 epochs after the network has reached convergence and stability. We illustrate the results in Fig. 4 and tabulate them in Table 3. Note that the competitive models are considered as a multi-class model and use categorical cross-entropy as the loss function. Since these models have different optimization functions for their own, we select Adam as the optimization function for these models in this work. Adam algorithm is frequently applied into neural networks and is generally the most stable one [52].

Fig. 4 demonstrates the three metrics of the accuracy, macro F1 score and micro F1 score with 95% confidence interval for different models. Specifically, Fig. 4(a) indicates the performance of models displaying Frequency, Time, TD, RawFrequency, TDFrequency, DeepSense, and IDNet with 2-second evaluation data. In particular, the TDFrequency model shows the highest accuracy and micro F1. Fig. 4(b) demonstrates the performance of the models with 5-second evaluation data. In this figure, for all the three metrics, the TDFrequency model shows the best performance. Therefore, our two-stream CNN achieves the highest accuracy.

Table 3. Comparison of accuracy, macro F1, and micro F1 (%) with standard deviation (SD) between different models over different time windows.

Time	Model	Accuracy (SD)	Macro F1 (SD)	Micro F1 (SD)
2 seconds	TD	71.51 (0.22)	64.23 (0.19)	71.51 (0.22)
	Time	83.18 (0.17)	77.52 (0.24)	83.18 (0.17)
	Frequency	86.05 (0.23)	81.25 (0.40)	86.05 (0.23)
	RawFrequency	87.14 (0.25)	82.99 (0.24)	87.14 (0.25)
	TDFrequency	87.54 (0.09)	85.29 (0.20)	87.54 (0.09)
	DeepSense	79.32 (1.33)	73.05 (1.54)	79.32 (1.33)
	IDNet	38.95 (0.57)	37.26 (0.57)	38.95 (0.57)
5 seconds	TD	70.98 (0.12)	63.44 (0.17)	70.98 (0.12)
	Time	82.29 (0.13)	76.27 (0.21)	82.29 (0.13)
	Frequency	88.75 (0.49)	81.62 (0.81)	88.75 (0.49)
	RawFrequency	90.01 (0.39)	84.43 (0.68)	90.01 (0.39)
	TDFrequency	90.04 (0.34)	85.05 (0.68)	90.04 (0.34)
	DeepSense	N/A	N/A	N/A
	IDNet	45.94 (0.43)	45.20 (0.67)	45.94 (0.43)

Table 3 lists the comparison results of accuracy, macro F1, and micro F1 with standard deviation (SD) between different models over different time windows. As depicted in Table 3, the TDFrequency model achieves the highest accuracy in both 2 and 5-second time windows. For the 2-second time window, the TDFrequency model achieves 87.54%, 85.29% and 87.54% with the standard deviation of 0.09%, 0.20% and 0.09% for accuracy, macro F1 and micro F1, respectively. Specifically, TDFrequency outperforms RawFrequency with margins of 0.4%, 2.3% and 0.4%, respectively, and with lower standard deviations. The results indicate that the temporal difference data combined with the frequency domain data are more efficient and accurate than the time domain data with it in the two-stream CNN. However, as a single input of CNN, the temporal difference data are not more accurate than the time domain data, because the time domain data include better temporal dynamic patterns. In addition, The TD model shows higher values with margins of 11.76%, 13.29% and 11.76% than the Time model. The Frequency model outperforms the Time model with margins of 2.87%, 3.73% and 2.87% for accuracy, macro F1 and micro F1, respectively. In addition, our model and the variants of Frequency, Time and RawFrequency outperform DeepSense model with margins of 3.86%, 4.47% and 3.86% at least. The IDNet model shows the worst performance compared with all the other models. Therefore, TDFrequency reaches the best performance in the 2-second time window.

For the 5-second time window in Table 3, the TDFrequency model achieves 90.04%, 85.05% and 90.04% with the standard deviation of 0.34%, 0.68% and 0.34% for accuracy, macro F1 and micro F1, respectively. The performance of the models with 5-second data is similar to that with 2-second data. However, The Frequency, TimeFrequency and TDFrequency models with 5-second data perform better than that in 2-second data while the Time and TD models with 5-second data are inferior to that in 2-second data. The results indicate that the increase of the frequency domain data improves accuracy. Since the experiment on DeepSense in the 5-second time window can not be trained on the GPU under our existing experimental conditions, the corresponding experimental results (accuracy, macro F1 score and micro F1) are not available. IDNet with 5-second data outperforms itself with 2-second data with margins of 6.99%, 7.94% and 6.99% for accuracy, macro F1 and micro

Table 4. Comparison of model size and computational cost over different time windows.

Time	Model	Parameter	MAdds
2 seconds	TD	0.99M	30M
	Time	0.99M	30M
	Frequency	0.99M	50M
	RawFrequency	1.8M	120M
	TDFrequency	1.8M	120M
	DeepSense	68.6M	330M
	IDNet	1.93M	20M
5 seconds	TD	2.17M	70M
	Time	2.17M	70M
	Frequency	2.17M	190M
	RawFrequency	4.16M	310M
	TDFrequency	4.16M	310M
	DeepSense	427.1M	840M
	IDNet	6.35M	80M

F1, respectively. This is likely because more authentication data contain more user information. Therefore, TDFrequency achieves the best performance in the 5-second time window.

6.2.2 Model size and computational cost. Table 4 lists the model comparison in terms of the model size and computational cost, where the model size refers to the parameters of the models, and the computational cost refers to the number of multiplication-addition operations (MAdd). We attempt to trade off the model parameters and the computational cost.

As illustrated in Table 4, TDFrequency, RawFrequency and its three variants of TD, Time and Frequency models are superior to the compared models of DeepSense and IDNet in the model size and partially in the computational cost. Specifically, in the 2-second time window, the TD and Time models achieve the best performance with 0.99 million parameters and 30 million MAdds. The Frequency model has a higher computational cost of 50M MAdds. As found, the RawFrequency and TDFrequency models both use 1.8M parameters and have a computational cost of 120M MAdds. Combined with the accuracy performance in Table 3, the TDFrequency model is comprehensively optimal, and thus we select TDFrequency model as the two streams inputs for the CNN. The accuracy of DeepSense model is comparable to the Time model; however, it is about 68 times larger in the model sizes and 11 times higher in the computational cost. Although IDNet has the smallest computational cost of 20M, its has the lowest accuracy of 38.95%, macro F1 of 37.26% and micro F1 of 38.95%.

In the 5-second time window, both of the model parameter and computational cost increase, because the input data increase in size. As listed in Table 4, the model performance is generally the same to that in the 2-second time window. In particular, the TDFrequency model uses 4.16M parameters and has a computational cost of 310M MAdds. Note that for DeepSense model, it uses 427.1M parameters and has a computational cost of 840M MAdds. The reason is that DeepSense can not be trained on the GPU in the 5-second time window.

6.3 Performance of SCANet

To evaluate the performance of SCANet on our dataset, we first explore the impact of the feature number on SCANet in terms of EER, FAR, and FRR. Then, we compare the accuracy of SCANet to representative authentication algorithms with manually designed features, such as KRR, SVM,

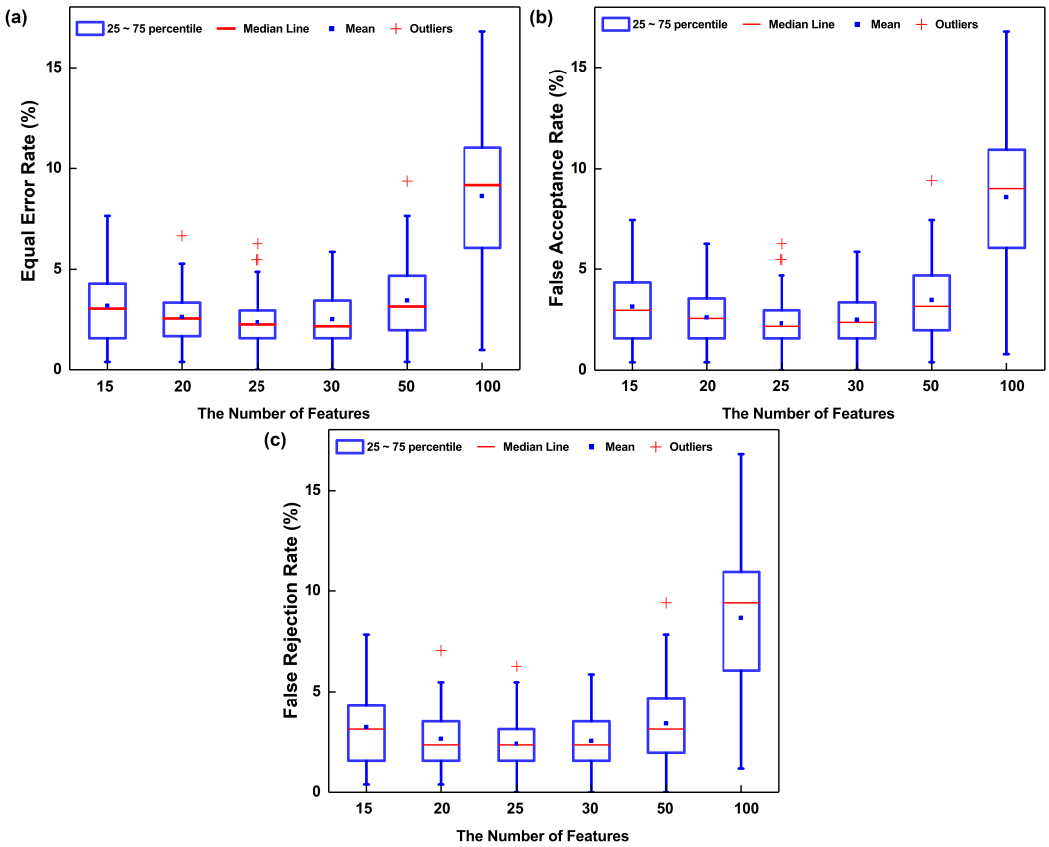

Fig. 5. EER, FAR and FRR for SCANet on different number of features.

one-class SVM, and *k*NN. Next, we compare the classifier performance of SCANet to comparable classifiers with CNN-extracted features. Finally, we evaluate the accuracy on unseen users and the time efficiency of SCANet, respectively. Note that the following experiments are conducted in a 2-second time window taking the accuracy and time cost into account. In addition, we introduce a public dataset BrainRun to further evaluate the performance of SCANet, and compare the performance between our dataset and BrainRun dataset.

6.3.1 Impact of the feature number. Fig. 5 depicts the box plots of EER, FAR, and FRR for SCANet on different number of features. As illustrated in Fig. 5(a), the EER first decreases with the increase of the selected features until 25, and then gradually increases. The FAR in Fig. 5(b) and FRR in Fig. 5(c) show the same trend to the EER. The results indicate that the feature number displaying the lowest EER, FAR, and FRR for SCANet is 25. In addition, Table 5 lists EER, FAR and FRR with standard deviation on different number of features. As shown in Table 5, based on 25 features, SCANet achieves the highest accuracy with 2.35% EER, 2.30% FAR and 2.40% FRR. Therefore, we choose 25 features with high discriminability by the PCA for SCANet in the feature extraction module.

6.3.2 Comparison of the accuracy. In order to verify the accuracy of SCANet, we compare the accuracy of SCANet to other representative algorithms, such as KRR, SVM, one-class SVM and

Table 5. EER, FAR and FRR (%) with standard deviation (SD) on different number of features.

# Feature	EER (SD)	FAR (SD)	FRR (SD)
15	3.16 (1.66)	3.11 (1.67)	3.22 (1.67)
20	2.61 (1.24)	2.58 (1.24)	2.64 (1.27)
25	2.35 (1.21)	2.30 (1.22)	2.40 (1.23)
30	2.50 (1.25)	2.46 (1.23)	2.54 (1.31)
50	3.42 (1.80)	3.43 (1.80)	3.41 (1.82)
100	8.59 (3.53)	8.55 (3.52)	8.62 (3.54)

Table 6. Manually designed features.

Feature	Explanation
Mean	Mean value of one-axis sensor readings
Standard Deviation	Standard deviation of one axis sensor readings
Maximum	Maximum value of one axis sensor readings
Minimum	Minimum value of one-axis sensor readings
Range	Difference between the maximum and minimum values
Kurtosis	Width of peak of one-axis sensor readings
Skewness	Orientation of peak of one axis sensor readings
Quartiles	25%, 50%, 75% quartiles of one axis sensor readings
Energy	Intensity of one axis sensor readings
Entropy	Dispersion of spectral distribution of one axis sensor readings
P1	Amplitude of the first highest peak of one axis sensor readings
P2f	Frequency of the second highest peak of one axis sensor readings
P2	Amplitude of the second highest peak of one axis sensor readings

Table 7. EER, FAR and FRR (%) with standard deviation (SD) on different algorithms.

Algorithm	EER (SD)	FAR (SD)	FRR (SD)
SCANet	2.35 (1.21)	2.30 (1.22)	2.40 (1.23)
KRR	9.31 (3.18)	9.31 (3.18)	9.32 (3.17)
SVM	13.90 (4.26)	13.91 (4.27)	13.90 (4.24)
One-class SVM	24.79 (6.06)	24.74 (6.07)	24.83 (6.06)
kNN	35.32 (6.44)	35.29 (6.43)	35.35 (6.45)

kNN with the same collected data. The features for the representative algorithms are manually designed, as shown in Table 6 [16, 23], where $2 \times 3 \times 15 = 90$ (2 sensors, 3 axis for each sensor, 15 features) features are used in total. As we described, SCANet exploits a single-class algorithm one-class SVM as its classifier and only requires users' own data for training and classifying. Note that SCANet and one-class SVM are different in that SCANet extracts features by the two-stream CNN and selects 25 features with high discriminability by the PCA, while one-class SVM just uses the manually designed features.

Fig. 6 illustrates the box plots of EER, FAR, and FRR for SCANet, KRR, SVM, One-class SVM and kNN with different features, respectively. As demonstrated in Fig. 6(a), SCANet has the lowest EER as compared to other representative algorithms. The FAR in Fig. 6(b) and FRR in Fig. 6(c) have the same trend to the EER. Table 7 lists EER, FAR, and FRR with standard deviation on the representative

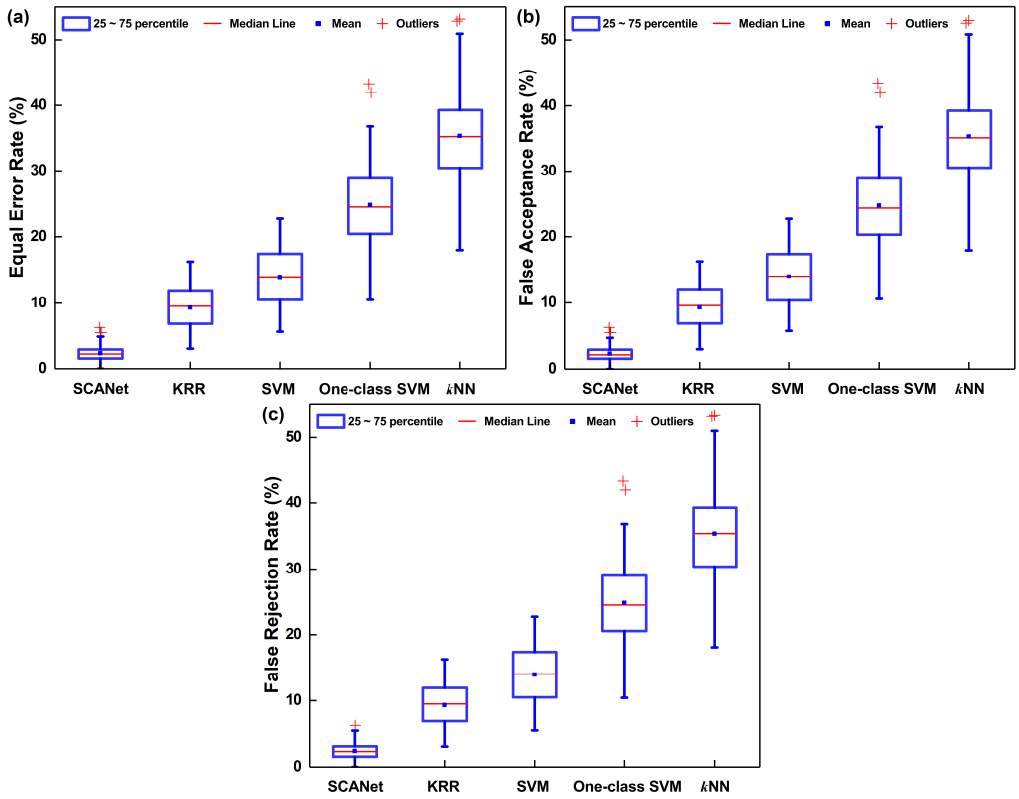

Fig. 6. Accuracy comparison between SCANet and representative algorithms with manually designed features.

algorithms. As demonstrated in Table 7, SCANet surpasses the representative algorithms with margins of 6.96%, 7.01% and 6.92% at least for the EER, FAR and FRR, respectively. In addition, the KRR demonstrates the highest accuracy among other representative algorithms, approximately reaching 9.31% EER, 9.31% FAR and 9.32% FRR.

6.3.3 Comparison of classifiers. To illustrate the efficiency of the one-class SVM with CNN-extracted features, we use the two-stream CNN to extract features for classifiers of one-class SVM, SVM, KRR and kNN. Fig. 7 describes the box plots of EER, FAR and FRR for SCANet and comparable classifiers with CNN-extracted features. In Table 8, the corresponding results of these classifiers are displayed. In this section, we refer to these classifiers as CNN-KRR, CNN-SVM, CNN-kNN. As depicted in Fig. 7, all the classifiers perform better when using CNN-extracted features. Specifically, the kNN achieves the best improvement, where the EER decreases from 35.32% (manually designed features) to 6.54% (CNN-extracted features), as shown in Table 8. The KRR and SVM outperform SCANet with margins of 0.6% EER, 0.6% FAR and 0.6% FRR. However, the KRR and SVM are two-class classifiers, which require positive and negative training data. The one-class SVM has the advantage of being able to use a very small positive training set to learn a classification function [53]. In addition, the computational complexity for the KRR and SVM is $O(N^3)$, while it is $O(N^2)$ for the one-class SVM [54]. The one-class SVM requires less training data and training time, and less storage space compared to the KRR and SVM. Therefore, we choose one-class SVM as the classifier for SCANet.

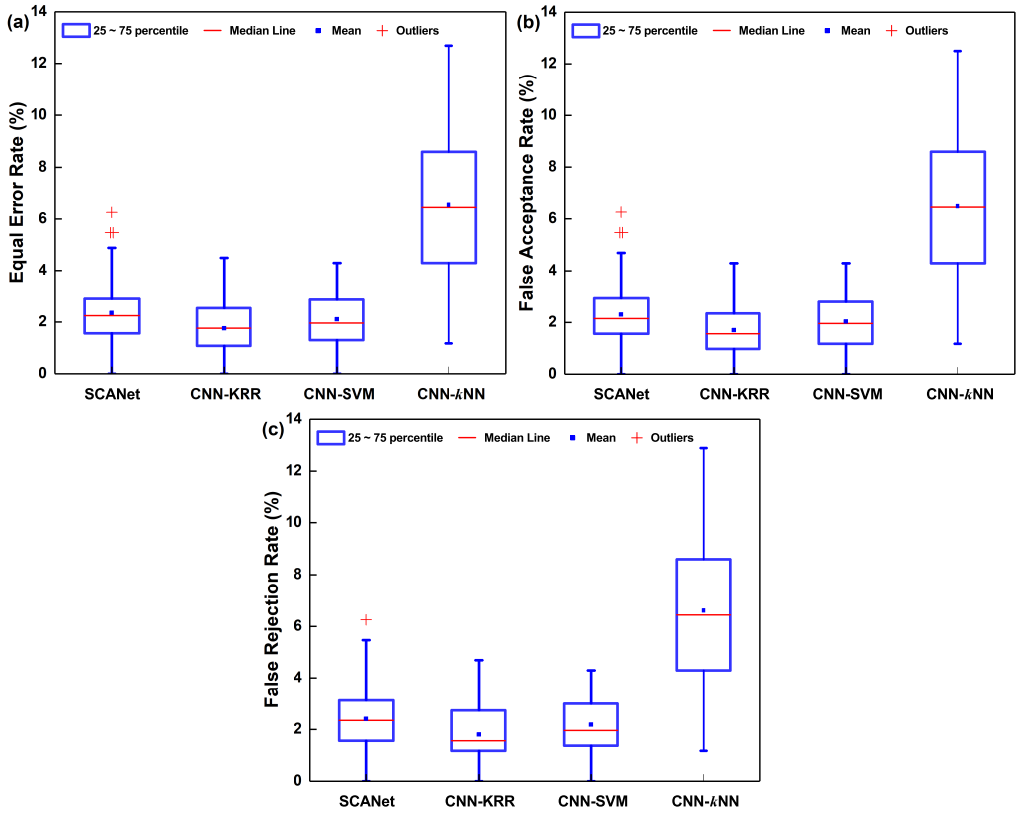

Fig. 7. EER, FAR and FRR of SCANet and comparable classifiers with CNN-extracted features.

Table 8. EER, FAR and FRR (%) with standard deviation (SD) on different algorithms.

Algorithm	EER (SD)	FAR (SD)	FRR (SD)
SCANet	2.35 (1.21)	2.30 (1.22)	2.40 (1.23)
CNN-KRR	1.75 (1.00)	1.70 (1.01)	1.80 (1.04)
CNN-SVM	2.10 (1.03)	2.02 (1.03)	2.18 (1.06)
CNN-KNN	6.54 (2.66)	6.47 (2.69)	6.61 (2.64)

6.3.4 Accuracy on unseen users. To show the performance of the pre-trained CNN on unseen users, we evaluate the accuracy of SCANet on unseen users. Based on our dataset containing 100 users, we randomly select n users whose data are used to train the two-stream CNN. Then, we exploit the pre-trained CNN to extract representative features from the rest (100-n) users whose data are not used in the CNN training. Based on the extracted features of the (100-n) users, we train and test the one-class SVM classifier in the same procedure as the second stage in Sec. 5.2, where we specify one of the (100-n) users as the legitimate user and the rest (100-n-1) as impostors. To generalize the accuracy, we set unseen users ($100 - n$) as 20, 30, 40, 50, 60, and 70, respectively. Fig. 8 describes the box plots of the EER, FAR and FRR for SCANet on different number of unseen users. As demonstrated in Fig. 8(a), the EER varies with the increase of the unseen user number from 20 to 70 (less than 9%) and shows lower mean values with 30, 40 and 50 unseen users. The FAR in

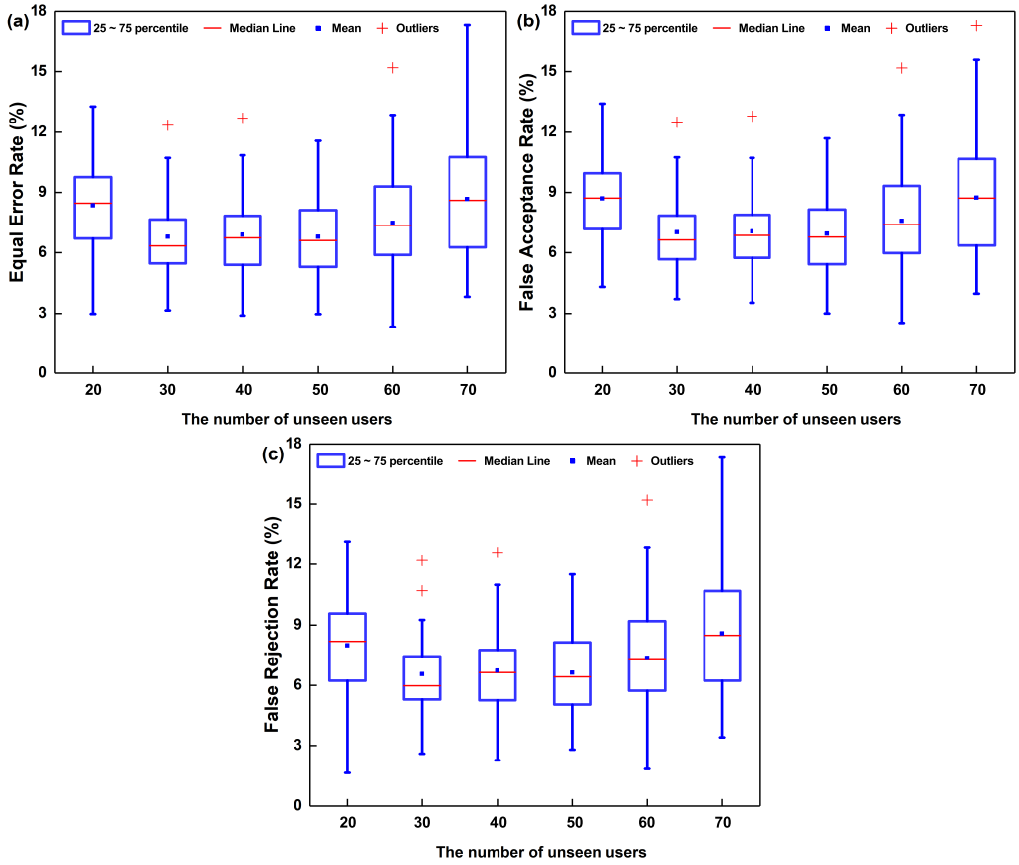

Fig. 8. EER, FAR and FRR for SCANet on different number of unseen users.

Table 9. EER, FAR and FRR (%) with standard deviation (SD) on different number of unseen users.

Unseen users	20	30	40	50	60	70
EER(SD)	8.34 (2.41)	6.78 (1.99)	6.89 (1.97)	6.78 (2.06)	7.47 (2.63)	8.66 (2.87)
FAR(SD)	8.69 (2.15)	7.00 (1.89)	7.05 (1.87)	6.93 (1.98)	7.57 (2.57)	8.73 (2.80)
FRR(SD)	7.99 (2.68)	6.56 (2.10)	6.73 (2.09)	6.63 (2.14)	7.36 (2.70)	8.58 (2.94)

Fig. 8(b) and FRR in Fig. 8(c) show the same trend to the EER. Moreover, Table 9 depicts the EER, FAR and FRR with standard deviation for SCANet on different number of unseen users. As listed in Table 9, when we select 70 users to train the two-stream CNN, and use the rest 30 users to train and test the one-class SVM classifier, SCANet achieves the best accuracy with 6.78% EER, 7.00% FAR and 6.56% FRR. With 50 unseen users, SCANet reaches almost the same accuracy with 6.78% EER, 6.93% FAR and 6.63% FRR, but the standard deviations for the EER, FAR, and FRR are slightly higher. On the other hand, with 70 unseen users, SCANet achieves the lowest accuracy with 8.66% EER, 8.73% FAR and 8.58% FRR, but they are all below 9%.

6.3.5 Time efficiency. The time cost of SCANet consists of: the length of a time window for authenticating (t_1), the time of feature extraction in the enrollment phase (t_2), authentication time

Table 10. Comparison with continuous authentication methods.

Method	Sensor	Feature	Participant	Classifier	Authentication	
					EER	Time
SCANet	acc., gyr.	CNN-extracted features	100	one-class SVM	2.35%	~3s
Roy *et al.* (2015) [22]	acc., gyr.	Touch-screen features	42	HMM	12.85%	N/A
Sitová *et al.* (2016) [23]	acc., gyr., mag.	HMOG features	100	Scaled Manhattan	7.16%	~60s
Shen *et al.* (2018) [25]	acc., gyr., mag., ori.	Descriptive and intensive features	102	HMM	4.74%	~8s
Li *et al.* (2019) [26]	acc., gyr.	Time and frequency features	100	one-class SVM	4.66%	~5s

in continuous authentication stage (t_3) and others (t_4), such as data preprocessing and system delay. In our experiments, taking the accuracy and model complexity into account, we choose a 2-second time window for authentication ($t_1 = 2s$). The feature extraction time and authentication time are $169ms$ and $1ms$ ($t_2 = 169ms$, $t_3 = 1ms$), respectively, for one sample. For other factors, different system environment presents some difference, so we ignore the time ($t_4 = 0$). The overall time cost is approximately 3 seconds, which is acceptable for interaction between a user and SCANet.

6.3.6 Comparison with authentication methods. We compare SCANet with four representative continuous authentication methods as illustrated in Table 10. As demonstrated in Table 10, we list the sensors for data collection, selected features, participants' number in the experiments, classifiers, and authentication performance in the EER and time. In the table, Roy *et al.* [22] and Sitová *et al.* [23] utilize the designed touch-screen features and HMOG features to train the Hidden Markov Model (HMM) and scaled Manhattan classifiers, respectively, but both reach low accuracy (12.85% EER in [22] and 7.16% for walk in [23]). Moreover, Shen *et al.* [25] and Li *et al.* [26] utilize the designed statistical features to train the HMM and one-class SVM classifiers, respectively, and achieve low accuracy with 4.74% in [25] and 4.66% in [26], but have long authentication time around 8s and 5s, respectively.

However, based on the temporal difference and frequency data, SCANet exploits the two-stream CNN to learn and extract representative features with high discriminability to achieve a 2.35% EER on our dataset and takes approximately 3s for user authentication.

6.4 Performance on BrainRun Dataset

To further evaluate the performance of SCANet, we select a publicly available dataset - BrainRun [55], which collected users' behavioral biometrics, such as touch gesture and motion sensor data from accelerometer and gyroscope. The dataset is mainly composed of two parts: gesture data and sensor data, where the sensor data in original dataset consist of accelerometer, gyroscope, magnetometer and deviceMotion sensors. In our experiment, we just use the accelerometer and gyroscope sensor data from deviceMotion sensors, which are collected by a called DeviceMotion library.

As the same selection criteria to our dataset, we selected 82 users from the BrainRun dataset, and chose 5120 seconds of data for each user for the experiments, where the 82 users' player_ids are tabulated in Table 11. Different from our dataset sampling rate of $100Hz$, the sampling rate of BrainRun dataset is $10Hz$, which indicates that the input shapes of the two-stream CNN are different. Due to the inputs with different shapes, the features in a certain layer can reduce to 0.

Table 11. player_id of the 82 users selected from BrainRun dataset.

r6ei6zy	tuvemg	ioxyr9y	uui53he	3ypgstx	rnhwbot	w0lcyp8
otu8x88	z8taipm	o16roh6	8uohdv4	8xjh8a	qa94pwy	d5deark
fit126a	hqqo4w4	o6g2zef	vo441ru	ta0ko40	hw7rn6q	h2o2ieg
7stgp0r	pxipu7n	d99p79w	9gykp6g	06mdn3c	60u0i9n	z3u2vz
txxrlzc	ldg6zjk	ntmyzk2	3pmv9aq	b0602fr	x8rbf3x	w8f2wrs
meakc9c	l1doqao	3zd4fdk	508rk86	yqhebyt	rcuqzrh	7nv0i9p
agmp5h7	n0g0zsv	lej2hfo	w764hxe	i6298h7	33a3a5u	r2p4ljs
7ksck80	uzip5ke	y2opfq8	k9ptsb2	lvrishm	lm8ujtt	bqwut29
6jtbpdh	0sqy7ba	vk43uke	xrm6gjj	56zi3vy	0z9xv36	68n9ll
l3rcqx4	677l8bq	cn1lmwf	ftk8v41	szmkl5t	n7ml8hx	r13q05q
aqq25vq	148r1k5	ydxu25a	gzx7rv	sxvkh3b	87g7ege	mixtfy4
9gx7uks	flidyt	frl4fzq	z8c5rtb	v90gnf6		

Therefore, we slightly adjust the input shapes, and the corresponding hyper parameters (stride, kernel, and avgpool) of the two-stream CNN architecture in Table 2 are reassigned values as: for individual convolutional subnet 1, the input shape is $(1, 2, 120)$, the stride is $(1, 2)$, and the kernel is $(1, 3 \times 1)$. For individual convolutional subnet 2, the input shape is $(1, 2, 60)$, the stride is $(1, 2)$, and the kernel is $(1, 3 \times 1)$. For merged convolutional subnet, the kernels are $(1, 8 \times 2)$, $(1, 8 \times 2)$, $(1, 8 \times 1)$, respectively, and the avgpool is $(1, 5)$.

To validate the performance of SCANet, we compare SCANet to other representative algorithms, such as KRR, SVM, one-class SVM and kNN on BrainRun dataset, where the features for the representative algorithms are manually designed, as shown in Table 6 [16, 23]. Note that for all the algorithms, we conduct the same procedure as the second stage in Sec. 5.2 to train and test the classifiers. Moreover, to show the performance of SCANet on unseen users, we train it using different user distribution, represented as SCANet-82 and SCANet-60, respectively.

SCANet-82: use the 82 users to train the two-stream CNN, and then use the same 82 users to train and test the one-class SVM classifier.

SCANet-60: randomly select 60 users out of the 82 to train the two-stream CNN, and then exploit the rest 22 users (unseen users) to train and test the one-class SVM classifier.

We list the model size and computational cost in a 2-second time window for SCANet-82 and SCANet-60 in Table 12. As shown in Table 12, SCANet-82 uses 1.13M parameters and has a computational cost of 1.12M MAdds while SCANet-60 uses 40M parameters and has 40M MAdds computational cost.

Fig. 9 demonstrates the box plots of EER, FAR, and FRR for SCANet-82, SCANet-60, KRR, SVM, One-class SVM and kNN on BrainRun dataset, respectively. As illustrated in Fig. 9, both SCANet-82 and SCANet-60 perform better in EER, FAR, and FRR, comparing to the other representative algorithms. In particular, SCANet-82 shows the best accuracy among all the algorithms. Table 13 depicts EER, FAR, and FRR with standard deviation on the representative algorithms. As listed in Table 13, SCANet-82 outperforms SCANet-60 with margins of 4.98% EER, 5.17% FAR, and 4.80% FRR. Moreover, SCANet on BrainRun dataset surpasses the representative algorithms with margins of 35.2%, 36.05%, and 34.34% at least for EER, FAR, and FRR, respectively.

6.5 Comparison on Different Datasets

To compare the performance of SCANet on our dataset and BrainRun dataset, we list the EER, FAR, and FRR with standard deviation on different datasets in Table 14. As listed in Table 14, in general,

Table 12. Model size and computational cost in a 2-second time window for SCANet-82 and SCANet-60.

Algorithm	Parameter	MAdds
SCANet-82	1.13M	1.12M
SCANet-60	40M	40M

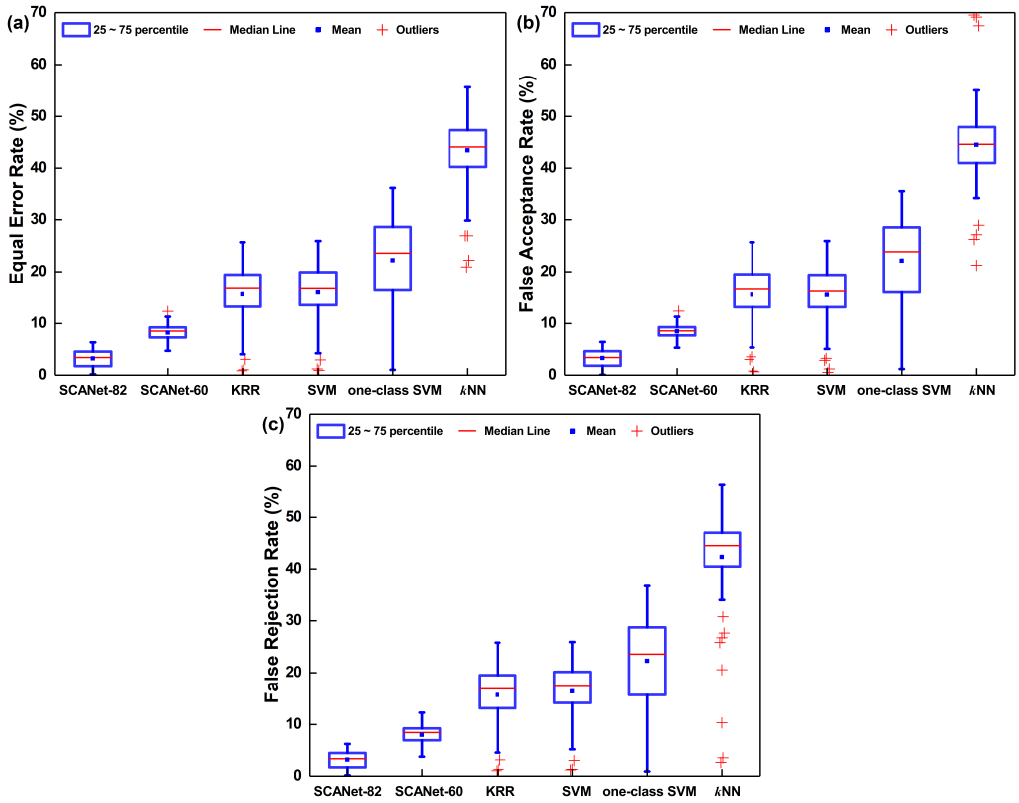

Fig. 9. Accuracy comparison between SCANet and representative algorithms with manually designed features on BrainRun dataset.

Table 13. EER, FAR and FRR (%) with standard deviation (SD) on different algorithms.

Algorithm	EER (SD)	FAR (SD)	FRR (SD)
SCANet-82	3.22 (1.68)	3.28 (1.67)	3.16 (1.68)
SCANet-60	8.20 (2.02)	8.45 (1.80)	7.96 (2.25)
KRR	15.61 (5.56)	15.53 (5.60)	15.69 (5.52)
SVM	15.95 (5.38)	15.48 (5.46)	16.41 (5.41)
One-class SVM	22.19 (8.34)	22.11 (8.35)	22.27 (8.35)
*k*NN	43.40 (6.43)	44.50 (7.72)	42.30 (9.52)

SCANet with our dataset performs better than that with BrainRun dataset. Specifically, SCANet on our dataset surpasses SCANet on BrainRun dataset with margins of 0.87% EER, 0.98% FAR,

Table 14. EER, FAR and FRR (%) with standard deviation (SD) on different datasets.

Algorithm	EER (SD)	FAR (SD)	FRR (SD)
SCANet on our dataset	2.35 (1.21)	2.30 (1.22)	2.40 (1.23)
SCANet on our dataset with 30 unseen users	6.78 (1.99)	7.00 (1.89)	6.56 (2.10)
SCANet on BrianRun dataset	3.22 (1.68)	3.28 (1.67)	3.16 (1.68)
SCANet on BrianRun dataset with 22 unseen users	8.20 (2.02)	8.45 (1.80)	7.96 (2.25)

and 0.76% FRR, respectively, which indicates that SCANet is strongly robust to different datasets. SCANet on our dataset with 30 unseen users outperforms that on BrainRun dataset with 22 unseen users with margins of 1.42% EER, 1.45% FAR, and 1.40% FRR, respectively, which represents SCANet has good performance on different datasets.

7 CONCLUSION

In this paper, we propose a novel two-stream CNN based authentication system, SCANet, for continuously authenticating users by leveraging their behavioral patterns. SCANet consists of five modules: data collection, data preprocessing, feature extraction, classification, and authentication. We design a two-stream CNN to learn and extract representative features and the most discriminable ones are further selected by the PCA. Then, with the CNN-extracted features, we use the one-class SVM to train the classifier in the enrollment phase. Based on the trained CNN and classifier, and testing features, SCANet classifies the current user as a legitimate user or an impostor in the continuous authentication phase. We evaluate the effectiveness of the two-stream CNN in terms of the accuracy, macro F1, micro F1, model parameters, and computational cost, and evaluate the performance of SCANet with respect to feature number impact, accuracy, classifiers, unseen users, time efficiency on our dataset, and the accuracy performance on BrainRun dataset, respectively. The experimental results show that the two-stream CNN achieves the best performance with an accuracy of 90.04%, macro F1 of 85.05%, micro F1 of 90.04%, and SCANet reaches an average EER of 5.14% on the two datasets and consumes approximately 3 seconds for user authentication, respectively.

ACKNOWLEDGMENTS

This work is partially supported by National Natural Science Foundation of China under Grants 61672119, 61762020, 61976030.

REFERENCES

[1] Arsalan Mosenia, Susmita Sur-Kolay, Anand Reghunathan and Niraj K. Jha. 2017. CABA: Continuous Authentication Based on BioAura. *IEEE Transactions on Computers* 66, 5(2017), 759-772.

[2] Pin Shen Teh, Andrew Beng Jin Teoh, and Shigang Yue. 2013. A Survey of Keystroke Dynamics Biometrics. *The Scientific World Journal*, 2013, Article ID 408280, 24 pages.

[3] Koichiro Niinuma, Unsang Park and Anil K. Jain. 2010. Soft Biometric Traits for Continuous User Authentication. *IEEE Transactions on Information Forensics and Security* 5, 4(2010), 771-780.

[4] C. Sanchez-Avila and R. Sanchez-Reillo. 2005. Two different approaches for iris recognition using gabor filters and multiscale zero-crossing representation. *Pattern Recognition* 38, 2(2005), 231-240.

[5] Lin Hong and Anil Jain. 1998. Integrating faces and fingerprints for personal identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 12(1998), 1295-1307.

[6] Ivan Martinovic, Kasper Rasmussen, Marc Roeschlin, and Gene Tsudik. 2017. Authentication using pulse-response biometrics. *Communications of The ACM* 60, 2(2017), 108-115.

[7] Frederic Bimbot, Jean-Francois Bonastre, Corinne Fredouille, et. al. 2004. A tutorial on text-independent speaker verification. *EURASIP Journal on Advances in Signal Processing* 2004, pp. 430-451.

[8] Huan Feng, Kassem Fawaz, and Kang G. Shin. 2017. Continuous authentication for voice assistants. In *Proceedings of the 23rd ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*. ACM/IEEE, 343-355.

[9] Shanxun Chen, Amit Pande, and Prasant Mohapatra. 2014. Sensor-assisted facial recognition: An enhanced biometric authentication system for smartphones. *Proceedings of the 12th annual international conference on Mobile systems, applications, and services (MobiSys)*, ACM, 109-122.

[10] Manuele Bicego, Andrea Lagorio, Enrico Grosso, and Massimo Tistarelli. 2006. On the use of sift features for face authentication. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*. IEEE, 1-7.

[11] Cheng Bo, Lan Zhang, Taeho Jung, Junze Han, Xiang-Yang Li and Yu Wang. 2014. Continuous user identification via touch and movement behavioral biometrics. In *Proceedings of IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, IEEE, 1-8.

[12] Bin Zou and Yantao Li. 2018. Touch-based smartphone authentication using import vector domain description. In *Proceedings of the 29th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, IEEE, 85-88.

[13] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S.-M. Makela, and H. Ailisto. 2005. Identifying users of portable devices from gait pattern with accelerometers. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE, II973-II976.

[14] Yantao Li, Hailong Hu, Gang Zhou, and Shaojiang Deng. 2018. Sensor-based continuous authentication using cost-effective kernel ridge regression. *IEEE Access* 6, (2018), 32554-32565.

[15] Senaka Buthpitiya, Ying Zhang, Anind K. Dey and Martin Griss. 2011. n-gram geo-trace modeling. In *Proceedings of International Conference on Pervasive Computing*, Lecture Notes in Computer Science, 97-114.

[16] Ge Peng, Gang Zhou, David T. Nguyen, Xin Qi, Qing Yang, and Shuangquan Wang. 2017. Continuous authentication with touch behavioral biometrics and voice on wearable glasses. *IEEE Transactions on Human-Machine Systems* 47, 3(2017), 404-416.

[17] Mario Parreño Centeno, Yu Guan and Ada van Moorsel. 2018. Mobile Based Continuous Authentication Using Deep Features. In *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning (EMDL)*. IEEE, 19-24.

[18] Mario Parreño Centeno, Ada van Moorsel and Stefano Castruccio. 2017. Smartphone Continuous Authentication Using Deep Learning Autoencoders. In *Proceedings of 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 147-155.

[19] Chris Xiaoxuan Lu, Bowen Du, Peijun Zhao, Hongkai Wen, Yiran Shen, Andrew Markham and Niki Trigoni. 2018. DeepAuth: In-situ Authentication for Smartwatches via Deeply Learned Behavioural Biometrics. In *Proceedings of the International Symposium on Wearable Computers (ISWC)*. ACM, 204-207.

[20] Upal Mahbub, Vishal M. Patel, Deepak Chandra, Brandon Barbello, and Rama Chellappa. 2016. Partial face detection for continuous authentication. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, 2991-2995.

[21] A. Hadid, J. Y. Heikkila, O. Silven, and M. Pietikainen. 2007. Face and eye detection for person authentication in mobile phones. In *Proceedings of the ACM/IEEE International Conference on Distributed Smart Cameras*. ACM/IEEE, 101-108.

[22] Aditi Roy, Tzipora Halevi, and Nasir Memon. 2015. An HMM-based multi-sensor approach for continuous mobile authentication. In *Proceedings of 2015 IEEE Military Communications Conference (MILCOM)*, 1311-1316.

[23] Zdeňka Sitová, Jaroslav Šeděnka, Qing Yang, Ge Peng, Gang Zhou, Paolo Gasti, and Kiran S. Balagani. 2016. Hmog: New behavioral biometric features for continuous authentication of smartphone users. *IEEE Transactions on Information Forensics and Security* 11, 5(2016), 877-892.

[24] Nabilah Shabrina, Tsuyoshi Isshiki, and Hiroaki Kunieda. 2016. Fingerprint authentication on touch sensor using phase-only correlation method. In *Proceedings of the 7th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*. IEEE, 85-89.

[25] Chao Shen, Yuanxun Li, Yufei Chen, Xiaohong Guan, and Roy A. Maxion. 2018. Performance Analysis of Multi-Motion Sensor Behavior for Active Smartphone Authentication. *IEEE Transactions on Information Forensics and Security*, 13, 1(2018), 48-62.

[26] Yantao Li, Hailong Hu, and Gang Zhou. 2019. Using data augmentation in continuous authentication on smartphones. *IEEE Internet of Things Journal* 6, 1(2019), 628-640.

[27] Oren Rippel, Jasper Snoek, and Ryan P. Adams. 2015. Spectral representations for convolutional neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*. ACM, 2449-2457.

[28] Tao Feng, Jun Yang, Zhixian Yan, Emmauenl M. Tapia, and Weidong Shi. 2014. Tips: contextaware implicit user identification using touch screen in uncontrolled environments. In *Proceedings of the 15th International Workshop on Mobile Computing Systems and Applications (HotMobile)*. ACM, 1-6, 2014.

[29] Hailong Hu, Yantao Li, Zhangqian Zhu, and Gang Zhou. 2018. CNNAuth: Continuous Authentication via Two-Stream Convolutional Neural Networks. In *Proceedings of 2018 IEEE 13th International Conference on Networking, Architecture*

and Storage (NAS). IEEE, 1-9.
[30] Tim Cooijmans, Nicolas Ballas, Cesar Laurent, caglar Gulcehre and Aaron Courville. 2017. Recurrent batch normalization. *arXiv Preprint arXiv:1603.09025 (2017)*
[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 770-778.
[32] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. 2018. Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types, *Computer-Aided Civil and Infrastructure Engineering* 33, (2018), 731-747.
[33] Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk. 2017. Deep Learning-Based Crack Damage DetectionU sing Convolutional Neural Networks, *Computer-Aided Civil and Infrastructure Engineering* 32, (2017) 361-378.
[34] Alex Krizhevsky, Ilya Sutskever, and Geofrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*. ACM, 1097-1105.
[35] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations (ICLR)*, 1-14.
[36] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv Preprint arXiv:1704.04861* (2017).
[37] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2017. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. *arXiv Preprint arXiv:1707.01083* (2017).
[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. In *Porceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 4510-4520.
[39] Karen Simonyan and Andrew Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*, ACM, 568-576.
[40] Laurent Slfre and Stephane Mallat. 2014. Rigid-motion scattering for texture classification. *International Journal of Computer Vision* 3559, (2014), 501-515.
[41] Francois Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Porceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 1800-1807.
[42] Lingjun Li, Xinxin Zhao, and Guoliang Xue. 2013. Unobservable re-authentication for smartphones. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS)*, 1-16.
[43] Cheng Bo, Lan Zhang, Xiang-Yang Li, Qiuyuan Huang, and Yu Wang. 2013. SilentSense: silent user identification via touch and movement behavioral biometrics. In *Proceedings of the 19th Annual International conference on Mobile Computing and Networking (MobiCom)*, ACM, 187-190.
[44] Larry M. Manevitz and Malik Yousef. 2002. One-class svms for document classification. *Journal of Machine Learning Research* 2, 2(2002), 139-154.
[45] Yantao Li, Bin Zou, Shaojiang Deng and Gang Zhou. 2020. Using Feature Fusion Strategies in Continuous Authentication on Smartphones. *IEEE Internet Computing*, DOI: 10.1109/MIC.2020.2971447.
[46] Wei-Han Lee and Ruby Lee. 2016. Implicit sensor-based authentication of smartphone users with smartwatch. In *Proceedings of the Hardware and Architectural Support for Security and Privacy (HASP)*, 1-8.
[47] Senaka Buthpitiya, Ying Zhang, Anind K. Dey, and Martin L. Griss. 2011. n-gram geo-trace modeling. In *Porceedings of International Conference on Pervasive Computing (ICPC)*, IEEE, 97-114.
[48] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, 351-360.
[49] Matteo Gadaleta, and Michele Rossi. 2018. IDNet: Smartphone-based gait recognition with convolutional neural networks. *Pattern Recognition* 74, (2018), 25-37.
[50] Chao Shen, Tianwen Yu, Sheng Yuan, Yunpeng Li, and Xiaohong Guan. 2016. Performance analysis of motion-sensor behavior for user authentication on smartphones. *Sensors* 16, 3(2016), 345:1-21.
[51] Leif E. Peterson. 2009. K-nearest neighbor. *Scholarpedia* 4, 2(2009), 1883. http://www.scholarpedia.org/article/K-nearest_neighbor
[52] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*. 1-15.
[53] Alexander Senf, Xue-wen Chen and Anne Zhang. 2006. Comparison of One-Class SVM and Two-Class SVM for Fold Recognition. In *Proceedings of 2006 International Conference on Neural Information Processing (ICONIP)*. Lecture Notes in Computer Science, 140-149.

[54] Pei-Yuan Wu, Chi-Chen Fang, Jien Morris Chang, and Sun-Yuan Kung. 2016. Cost-Effective Kernel Ridge Regression Implementation for Keystroke-Based Active Authentication System. *IEEE Transactions on Cybernetics* 47, 11(2017), 3916-3927.

[55] Michail D. Papamichail, Kyriakos C. Chatzidimitriou, Thomas Karanikiotis, Napoleon-Christos I. Oikonomou, Andreas L. Symeonidis, and Sashi K. Saripalle. 2019. BrainRun: A Behavioral Biometrics Dataset towards Continuous Implicit Authentication. *Data* 42, (2019), 1-17.

Received February 2019; revised February 2020; accepted April 2020