CNNAuth: Continuous Authentication via Two-stream Convolutional Neural Networks

Hailong Hu
College of Computer and Information Sciences
Southwest University
Chongqing 400715, China

Yantao Li*
College of Computer Science
Chongqing University
Chongqing 400044, China
*yantaoli@cqu.edu.cn

Zhangqian Zhu
College of Computer and Information Sciences
Southwest University
Chongqing 400715, China

Gang Zhou
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA

***Abstract*—We present a two-stream convolutional neural network based authentication system, *CNNAuth*, for continuously monitoring users' behavioral patterns, by leveraging the accelerometer and gyroscope on smartphones. We are among the first to exploit two streams of the time-domain data and frequency-domain data from raw sensor data for learning and extracting universal effective and efficient feature representations as the inputs of the convolutional neural network (CNN), and the extracted features are further selected by the principal component analysis (PCA). With these features, we use the one-class support vector machine (SVM) to train the classifier in the enrollment phase, and with the trained classifier and testing features, *CNNAuth* classifies the current user as a legitimate user or an impostor in the continuous authentication phase. We evaluate the performance of the two-stream CNN and *CNNAuth*, respectively, and the experimental results show that the two-stream CNN achieves an accuracy of 87.14%, and *CNNAuth* reaches the lowest authentication EER of 2.3% and consumes approximately 3 seconds for authentication.**

***Index Terms*—Continuous authentication, universal feature representations, convolutional neural network (CNN), one-class support vector machine (SVM), equal error rate (EER)**

I. Introduction

Current continuous authentication mechanisms on smartphones 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 [1], [2]. On the other hand, it is not easy to design an effective continuous system without limitations, such as latency, representational power of extracted features, and computational cost [3]–[6].

To address the first challenge, we generate two-stream data from the raw sensor data: time-domain data and frequency-domain data. The time-domain data are directly collected by smartphone sensors, which contain temporal movements that are crucial to explicitly capture the dynamic temporal features [7]. Frequency-domain data are converted from raw sensor data by Fourier transformation, which contain better local frequency patterns that not only alleviate the impact of noise but also are independent on how time-series data are organized in the time domain [8]. For the second challenge, we design a convolutional neural network (CNN) based on the two-stream data (two-stream CNN) to learn and extract the effective and efficient data representations. The two-stream CNN is adaptive to resource-constrained mobile devices through significantly decreasing network parameters and the number of operations while retains the authentication accuracy.

In this paper, we propose a novel two-stream convolutional neural network based authentication system, *CNNAuth*, for continuously authenticating users' behavioral patterns, leveraging the accelerometer and gyroscope on smartphones. More specifically, *CNNAuth* consists of six modules of the data collection, data preprocessing, feature extraction, feature selection, classifier, and authentication. The operation of *CNNAuth* includes the enrollment phase for data collection, learning universal features by training the two-stream CNN, and classifier training, and the continuous authentication phase for classifier testing and authentication. Data collection captures users' behavioral patterns during smartphone usage, by utilizing the two sensors of the accelerometer and gyroscope omnipresently built-in smartphones. Data preprocessing converts the raw sensor data into two stream of inputs for the CNN. In the feature extraction module, we specially design a two-stream CNN to learn and extract the effective and efficient data representations for resource-constrained mobile devices. Note that the two-stream CNN is trained only once on a representative user dataset in the preprocessing module, and then is used at runtime as a universal feature extractor. In feature selection module, the most discriminable ones among these features are extracted by the two-stream CNN and further selected by the principal component analysis (PCA) in feature selection module. Then, we use the one-class support vector machine (SVM) to train the classifier in the enrollment phase. With the trained classifier and testing features, *CNNAuth* 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 *CNNAuth* with respect to the equal error rate (EER) and time efficiency, respectively. The experimental results show that the two-stream CNN achieves best performance with accuracy of 87.14%, macro F1 of 82.99%, micro F1 of 87.14%, model parameters of 1.8M, and computational cost of 120M, and *CNNAuth* reaches the lowest authentication EER of 2.3% and consumes approximately 3 seconds for authentication, respectively.

In summary, the main contributions of this work are three-fold:

- We design a two-stream convolutional neural network based authentication system, *CNNAuth*, for continuously monitoring users' behavioral patterns, by leveraging the accelerometer and gyroscope on smartphones. *CNNAuth* is composed of the data collection, data preprocessing, feature extraction, feature selection, classifier, and authentication.
- We are among the first to exploit the two streams of the time-domain data and frequency-domain data from raw sensor data as the inputs of CNN for learning and extracting universal effective and efficient features.
- We evaluate the effectiveness of the two-stream CNN and the performance of *CNNAuth*, and the experimental results show that the two-stream CNN achieves the accuracy of 87.14%, and *CNNAuth* reaches the lowest authentication EER of 2.3% and consumes approximately 3 seconds for authentication.

The remainder of this paper is organized as follows: Sec. II reviews the-state-of-art in efficient network architectures and continuous authentication system, and Sec. III describes the two-stream architecture for learning effective and efficient representations. In Sec. IV, we detail the architecture of *CNNAuth* in the data collection, feature selection, classifier, and authentication. We describe the experiments on *CNNAuth* in Sec. V and evaluate the performance of two-stream CNN and *CNNAuth* in Sec. VI. We conclude this work in Sec. VII.

II. Related Work

This section reviews the state-of-art of efficient network architectures and continuous authentication systems in detail.

A. Efficient Network Architecture

Recently, neural networks have become one of the most popular methodologies in many areas of machine intelligence. A lot of efficient network architectures have been proposed to achieve superhuman accuracy, such as *AlexNet* [9], *VGGNet* [10], and *ResNet* [11]. 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 platform, such as smartphones. There are some works dedicated to tuning neural network architectures to reach an optimal trade-off between accuracy and performance on some mobile and embedded applications, such as *MobileNet* [12], *ShuffleNet* [13], and *MobileNetV2* [14]. 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 to consider sensor data inputs. This work differs in that our designed neural network architecture mainly deals with sensor or multi-sensor inputs, and at the same time, our system can be performed on the computationally limited platform, such as smartphones.

B. Continuous Authentication System

Most authentication mechanisms on smartphones, such as passwords, PINs, fingerprint, and facial identification, provide security just only by a one-pass session, which enables impostors to access the system until the user logs out. To address this problem, continuous authentication mechanisms are explored and developed, which generally can be categorized into two groups: physiological biometrics based approaches and behavioral biometrics based approaches. The physiological biometrics based approaches authenticate users by static physical attributes, such as the fingerprint, voice, and facial identification. In [15] the authors propose an application of the scale invariant feature transform approach in the context of the face authentication. The authors in [16] propose a continuous authentication system *VAuth* through executing only the commands that originate from the voice of the owner. However, these approaches require users direct participation. Behavioral biometrics based approaches authenticate users by the invariant features of human behaviors during different activities, such as touch gestures, and gait. In [17], the authors propose a touch-based authentication system by exploiting a novel one-class classification algorithm import vector domain description during smartphone usage. The author in [18] propose a novel sensor-based continuous authentication system, *SensorCA*, for continuously monitoring users behavior patterns, by leveraging the accelerometer, gyroscope, and magnetometer ubiquitously built-in smartphones. However, these approaches can not achieve better performance due to lacking of robust features or efficient algorithms for authentication. Our work differs in that we design a lightweight CNN to learn universal, efficient and roust features to achieve better performance for continuous user authentication.

III. Architecture of Two-stream CNN for Learning Effective and Efficient Representations

Learning effective and efficient data representations is critical to continuous authentication systems since the performance of real-word systems conforms to a set of criteria, such as the latency, representational power of extracted features, and inference speed of the feature extractor. In this section, we first describe how to convert the raw sensor data into two streams of inputs for CNN, which is the basis of the effective and efficient representations. Then, we introduce the depthwise separable convolutions and linear bottlenecks, which are crucial building

blocks for the two-stream CNN [19]. Finally, we elaborate the architecture of the two-stream CNN, which is designed for learning universal features with less network parameters and less operations while retains the authentication accuracy.

For the rest of this paper, all vectors are represented by bold lower-case letters (e.g., $\mathbf{x}$ and $\mathbf{y}$), and matrices and tensors are denoted by bold upper-case letters (e.g., $\mathbf{X}$ and $\mathbf{Y}$). Any element in vectors or tensors is represented by lower-case letters (e.g., x and y).

A. Data preprocessing

The synchronized raw sensor readings of the accelerometer and gyroscope are represented by a vector $\mathbf{w}= (x_{acc}, y_{acc}, z_{acc}, x_{gyro}, y_{gyro}, z_{gyro})^T \in R^6$, 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. In each time interval, a matrix $\mathbf{Q} = (\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w}_\tau)$ has a shape $d \times \tau$. We apply Fourier transformation to each element in $\mathbf{Q}$ in each time interval, and stack these frequency domain sensor data into a $d \times 2f$ matrix $\mathbf{X_f}$, where f is the dimension of frequency domain containing f magnitude and f phase pairs ($f = \tau$). For raw sensor readings (time-domain data) $\mathbf{X_t}$, we do not make any change. 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, and T is the number of time windows. The length of a time window usually determines the time that the system requires to perform the continuous authentication. In this work, we use time-domain data $\mathbf{X_t}$ and frequency-domain data $\mathbf{X_f}$ as our two streams of inputs for CNN.

B. Depthwise Separable Convolution and Linear Bottleneck

Depthwise Separable Convolutions were initially introduced in [20] and subsequently applied to Inception models [21] to reduce the computation cost. They generally factorize a standard convolution into a depthwise convolution for filtering and a pointwise convolution for combining. More specifically, a depthwise convolution performs lightweight filtering by applying a single filter on each input channel while a pointwise convolution builds new features by computing linear combinations of the input channels.

In data preprocessing (Sec. III-A), we show that sensor readings in a time window can be represented to a tensor with shape $T \times d \times \tau$ or $T \times d \times 2f$. Furthermore, we unify the representations with shape $c \times h \times w$ for convenience, where c is the number of time windows, h is the number of time intervals, and w is the dimension of each sensor readings for all the sensors. The reason is that we refer to the representation of an image $c \times h \times w$, where c is the number of channels, and h and w are the height and the width, respectively.

A standard convolution takes a $c \times h \times w$ input tensor $\mathbf{L}$, and then applies convolutional kernel $K \in R^{c' \times c \times k_1 \times k_2}$ to produce a $c' \times h \times w$ input tensor $\mathbf{L}'$. The computational cost of a standard convolutional layer is $h \times w \times c \times c' \times k_1 \times k_2$. However, depthwise separable convolutions empirically work as well as standard convolutions but only cost $\frac{k_1 \times k_2 \times c'}{k_1 \times k_2 + c'}$, which is the sum of the depthwise and pointwise convolutions. Compared with a standard convolutional layer, depthwise separable convolution reduces 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, we use 1×32 depthwise separable convolution for convolutional layers, so the computational cost is 32 times smaller than that of standard convolutions at the cost of a small reduction in accuracy [22]–[24].

Linear Bottlenecks were initially proposed by Mark Sandler [14] based on depthwise separable convolutions. For a block with size $h \times w$, kernel size $k_1 \times k_2$, expansion factor t, c input channels and c' output channels, it takes a low-dimensional input $c \times h \times w$. the block is first expanded to high dimension by expansion factor t and filtered with a lightweight depthwise convolution with kernel size $k_1 \times k_2$. Then its features are projected back to a low-dimensional output $c' \times h \times w$ with a linear convolution.

There are two reasons why our two-stream CNN is based on Linear Bottleneck blocks: 1) they provide a better understanding of our designed network since there is a natural separation between the input/output domains of the bottleneck layers and the layer transformation [14]; 2) they can reduce the number of parameters and computational cost in convolutional operations. In this work, we do not use skip connection because we set $stride = 1$. The total number of multiplications and additions required is $h \times w \times c \times t \times (c + k_1 \times k_2 + c')$.

C. Two-stream CNN Architecture

We illustrate the two-stream CNN architecture in Fig. 1. As discussed in previous sections, its basic building block is a linear bottleneck. It consists of an individual convolutional subnet for each input tensor (input tensors refer to $\mathbf{X_f}$ and $\mathbf{X_t}$), and a single merged convolutional subnet for the outputs of the two individual convolutional subnets. In this work, the network concentrates on learning effective and efficient representations by detecting spatial patterns that are related to the frequency domain sensor readings and local temporal dynamic patterns which are related to the raw sensor readings. Recall that frequency-domain data $\mathbf{X_f}$ and time-domain data $\mathbf{X_t}$ can be represented with a tensor $n \times T \times n \times 2f$ and a tensor $n \times T \times d \times \tau$, respectively, where n is the number of sensor data, T is the number of time windows, d is the dimension of all sensor data, f is the dimension of sensor data in frequency domain, and τ represents the number of sensor data in one time interval. They will be fed into the network as two-stream inputs. We extract 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 relationships across sensor data dimensions. The frequency domain generally includes many spatial patterns in some neighboring frequencies.

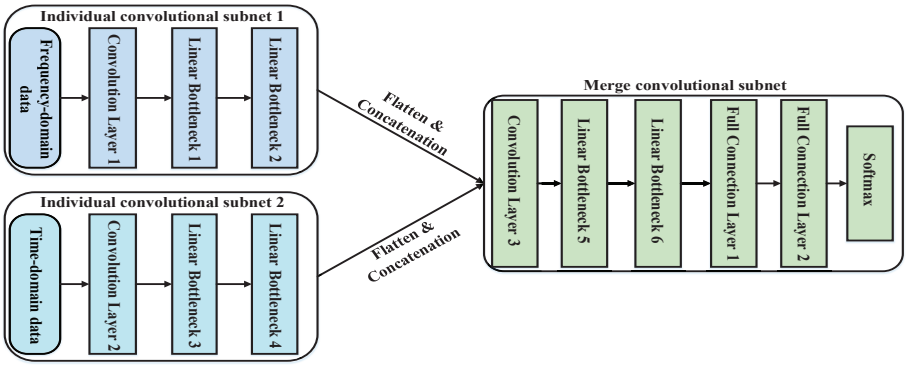

Fig. 1. Architecture of the two-stream CNN.

The temporal domain commonly contains a number of local temporal dynamic patterns. The interaction among sensor data usually contains all dimensions. Therefore, for $\mathbf{X_f}$, we first apply $2d$ filters with shape $(d, Conv1)$ to $\mathbf{X_f}$ to learn interaction among sensor data dimensions and spatial patterns in the frequency domain. Then we apply Linear Bottlenecks with shapes $(1, Conv2)$ and $(1, Conv3)$ hierarchically to learn high-level features. Similarly, for $\mathbf{X_t}$, the process is the same since the structure of the individual convolutional subnet is the same.

Then we flatten their outputs and concatenate them along channels. The structure of the merged convolutional subnet is similar to the individual convolutional subnet. We first apply $2d$ filters with shape $(2, Conv4)$ to learn the interactions between the temporal and frequency domains, and then apply Linear Bottlenecks with shapes $(1, Conv5)$ and $(1, Conv6)$ hierarchically to learn high-level features. Finally, we use two full connection layers, which are used to classify the input patterns into a finite number of classes. The output of the last full connection layer will be fed into a softmax layer to generate the predicted category probability.

For the individual convolutional subnet, two-stream CNN learns 32 filters, and for merged convolutional subnet, it learns 64 filters. 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 $32 \times 6 = 192$ channels. We use Rectified Linear Unit (ReLU6) as our activation function due to its robustness even in low-precision computation [12]. In addition, batch normalization is applied at each layer to reduce internal covariate shift, and dropout is also employed during training. 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.

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 multiplications and additions 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 (the number of multiplication-addition operations) 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.

IV. System Design

In this section, we present the two-stream convolutional neural network based user authentication system, *CNNAuth*, for continuously monitoring users' behavioral patterns by exploiting sensor data from the accelerometer and gyroscope on smartphones. The architecture of *CNNAuth* is illustrated in Fig. 2. As shown in Fig. 2, *CNNAuth* consists of six modules: data collection, data preprocessing, feature extraction, feature selection, classifier, and authentication. The operation of *CNNAuth* includes two phases for learning and classifying users' behavioral patterns: the enrollment phase and continuous authentication phase. In particular, the preprocessing module mainly focuses on learning effective and efficient representations, and the feature extraction module can automatically extract universal features by applying the two-stream CNN on training datasets, which we have discussed in Sec. III-C. In the remainder of this section, we elaborate the rest of modules: data collection, feature selection, classifier, and authentication.

A. Data collection

Data collection module collects all users' sensor data from the accelerometer and gyroscope. The accelerometer records a user's motion patterns, such as how to move the arms or how to walk, and the gyroscope records a user's fine-grained motions, such as how to hold smartphones during usage. The two sensors do not require root permission when requested by mobile applications, which makes them useful in background monitoring. In *CNNAuth*, the data collection module captures

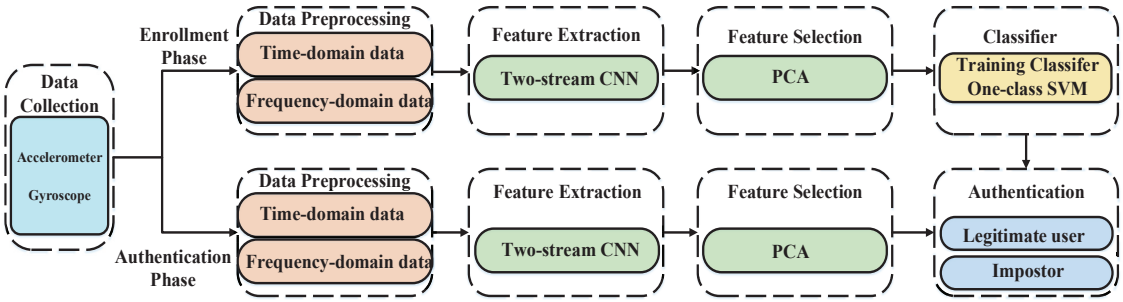

Fig. 2. Architecture of *CNNAuth*.

the user's every subtle movement during the operations on 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.

B. Feature selection

After data preprocessing and feature extraction as detailed in Sec. III-C, the extracted features are passed to feature selection module and features with high discriminability are selected. In *CNNAuth*, we exploit the principal component analysis (PCA) to select 25 features.

C. Classifier

The selected features are then fed to the classifier for training and testing, respectively. We implement the one-class support vector machine (SVM) classifier, which exploits a kernel function to map data into a high dimensional space, and considers the origin as the only sample from other classes [25]. 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.

D. Authentication

Based on the testing feature vectors 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, *CNNAuth* will require initial login inputs; otherwise, it will continuously authenticate the user.

V. Experiment

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

A. Dataset

To investigate the accuracy of *CNNAuth*, we collected sensor data from 100 smartphone users (53 male, and 47 female). To collect sensor data, each user devoted to approximately 2 to 6 hours of behavior traits including document reading, text production, and navigation on a map locating a destination [26]. We recorded sensor readings of the accelerometer and gyroscope with the sampling rate of $100Hz$, by an android-based software system. In this work, we select the first 100 minutes of data for each user with a 2-second window size.

B. Training

The training phase of *CNNAuth* 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 *CNNAuth*. The trained two-stream CNN is fixed as an universal feature extractor, and we just 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. That is, we have positive feature samples from one legitimate user and 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 the effect of the randomness.

C. Metrics

We explore four metrics that are used to analyze the authentication accuracy of *CNNAuth*: accuracy, Micro F1 score, Macro F1 score, and equal error rate (EER).

VI. Experimental results

In this section, we evaluate the effectiveness of the two-stream CNN and the performance of *CNNAuth*, respectively.

We have implemented our system *CNNAuth* by utilizing Python. The two-stream CNN and exiting algorithms, such as KRR, one-class SVM, and kNN, mainly call the Pytorch framework and sklearn package, respectively. 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 tests (testing two-stream CNN in the preprocessing module, training and testing authentication algorithms in the enrollment phase and in the continuous authentication phase) 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 algorithms or models for comparison.

A. Algorithms for comparison

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

1) Comparison models for two-stream CNN: We compare our two-stream CNN model with other competitive models. To show the direct difference, we refer to our model as RawFrequency-CNN. There are two variants of the two-stream CNN model by utilizing single-stream input: Raw-CNN and Frequency-CNN. The competitive models for comparison are:

Raw-CNN: raw sensor data are used as single-stream input. That is, this model has only one individual convolutional subnet, and there is no concatenation for the merged convolutional subnet [27].

Frequency-CNN: frequency data are used as single-stream input. That is, this model has only one individual convolutional subnet, and there is no concatenation for the merged convolutional subnet [28].

Deepsense: it integrates convolutional neural network and recurrent neural networks to authenticate user. The frequency representations of sensor data are fed into the model [29].

2) Comparison algorithms for CNNAuth: We compare *CNNAuth* with other competitive authentication algorithms. These authentication algorithms are: kernel ridge regression (KRR), one-class SVM, k-nearest neighbors (kNN), respectively:

KRR: the classifier is the combination ridge regression (linear least squares with l2-norm regularization) with 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 using the training vectors with the RBF kernel function, which is similar to a SVM. KRR parameters and kernel parameters are set by grid search [30]. In the continuous authentication phase, the classifier maps the testing vector by the RBF kernel function into the high-dimension space, and calculates the distance between the testing vector and the linear separator as the classification score.

One-class SVM: the classifier is regarded as an unsupervised learning algorithm, which projects data onto a high dimensional space through a kernel function, and regards the origin as only sample from other classes [31]. In the enrollment phase, the classifier is trained by using 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, and calculates the distance between the testing vector and the linear separator as the classification score. The difference between one-class SVM algorithm and *CNNAuth* is that *CNNAuth* uses two-stream CNN to preprocess the collected data, since *CNNAuth* exploits one-class SVM as the classifier.

kNN: the classifier authenticates a user through the assumption that the testing vector from user will resemble one or more of those in the training vectors [32]. 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 is used as the classification score.

B. Performance of two-stream CNN

In this section, to validate the effectiveness of the RawFrequency-CNN, we evaluate the performance by comparing with other competitive models with respect to the accuracy, computational cost, and model size, respectively.

1) Accuracy, macro F1 and micro F1: The two-stream CNN is considered as a multi-class classification problem (100 classes) and trained through 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. Note that the competitive models are considered as a multi-class model and use categorical cross-entropy as the loss function. Although these models have different optimization functions, we select Adam as optimization functions for these models in our work. The performance of Adam algorithm is generally the most stable one and it is applied frequently in neural networks [33].

Fig. 3 shows the three metrics of the accuracy, macro F1 score and micro F1 score with 95% confidence interval for different models. More specifically, Fig. 3(a) shows the results of models of Raw, Frequency, RawFrequency, and Deepsense with 2-second evaluation data. For the three matrics, RawFrequency shows the highest values. Fig. 3(b) shows the results of

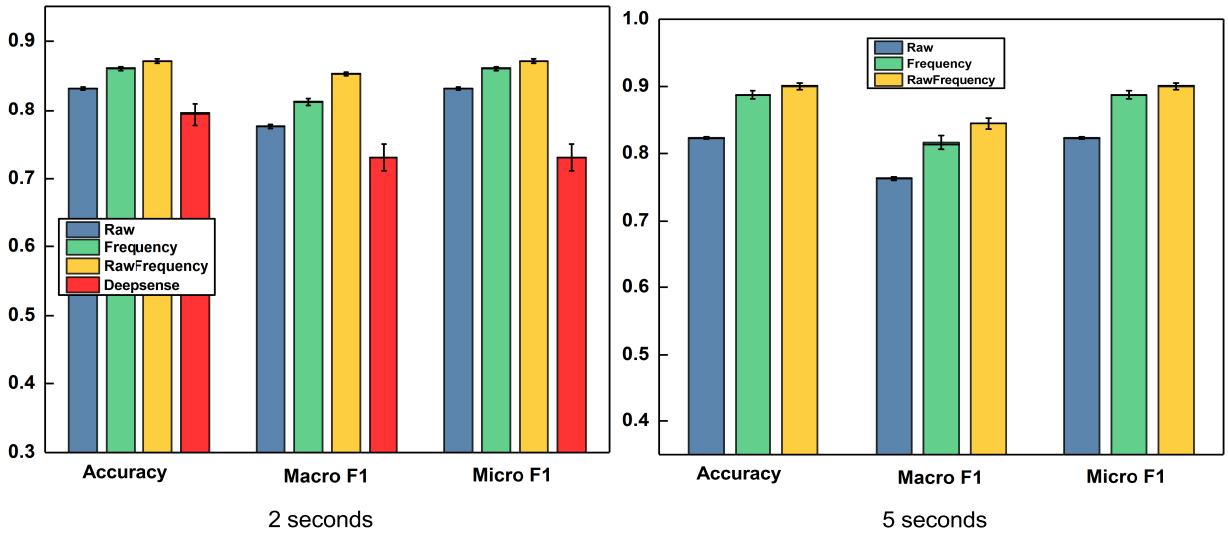

Fig. 3. Accuracy comparison results

models of Raw, Frequency, and RqwFrequency with 5-second evaluation data. For all the three matrics, RawFrequency shows the highest values as well. Therefore, our RawFrequency-CNN has the best accuracy.

Table I shows the comparison results of accuracy, macro F1, and micro F1 with standard deviation within parentheses between different networks over different time windows. As shown in Table I, the RawFrequency, Raw, and Frequency outperform the Deepsense model with a margin 3.86% at least. In comparison with RawFrequency-CNN and the two variants, our RawFrequency-CNN model is more efficient than the two single-stream CNN models. Comparing with Raw and Frequency models, frequency input based the CNN model achieves better performance. It validates that the frequency-domain data have good patterns and better resistance to noise while raw data as a single-stream input have a poor accuracy. However, comparing with frequency-domain data as two-stream inputs, Raw model achieves improvements. The accuracy of Deepsense is the worst in all models, and the accuracy in 5-second time window overall is higher than that in 2-second window. This is because more authentication data contain more user information. In summary, our RawFrequency-CNN (*CNNAuth*) achieves the best performance with accuracy of 87.14%, macro F1 of 82.99%, and micro F1 of 87.14%.

2) Model size and computational cost: Table II shows the model comparison in the model size and computational cost, where the model size refers to the parameters of the networks, and the computational cost refers to the number of multiplication-addition operations (MAdd). We try to trade off the model parameters and the computational cost.

As illustrated in Table II, our RawFrequency and its two variants Raw and Frequency are superior to the compared model Deepsense in terms of the model size and computational cost. More specifically, in a 2-second time window, Raw achieves the optimal performance, but combined with the accuracy in Table I, RawFrequency is comprehensively optimal. We finally choose RawFrequency as our two-stream CNN to extract features under the careful consideration, whose parameters are about 1.8M and MAdds are about 120M. Although the accuracy of Deepsense model is comparable to the Raw model, it is much larger than our model in terms of the model size and computational cost. Its parameters are about 68 times bigger and the 11 times more calculations than ours. In the 5-second time window, both the model parameters and computational cost for all the models increase, because the input data increase in size. However, the growth rate of our model is relatively small but still observable. Note that for Deepsense model, its parameters are about 427.1M and its MAdds are about 840M. The reason is that our experiment about Deepsense in the 5-second time window can not be trained on the GPU under our existing experimental conditions, so corresponding experimental results (accuracy, macro F1 score and micro F1) are not measured as shown "N/A" in Table I.

C. Performance of CNNAuth

To evaluate the performance of *CNNAuth*, we first explore the impact of number of features on the EER and time efficiency. Then, we compare *CNNAuth* with manually designed feature based common authentication algorithms, such as KRR, SVM, one-class SVM, and kNN.

As a continuous authentication system, *CNNAuth* uses a single-class algorithm one-class SVM as the classifier. It only requires user's own data for training and classifying. We compare the *CNNAuth* with traditional authentication algorithms based on manually designed features, such as traditionalFeature-oneClassSVM, traditionalFeature-SVM, traditionalFeature-KNN, and traditionalFeature-kRR. Note that the following experiments are conducted in a 2-second time window taking the accuracy and time into account.

TABLE I

COMPARISON RESULTS OF ACCURACY, MACRO F1, AND MICRO F1 WITH STANDARD DEVIATION WITHIN PARENTHESES BETWEEN DIFFERENT NETWORKS OVER DIFFERENT TIME WINDOWS.

Time	Network	Accuracy (%)	Macro F1 (%)	Micro F1 (%)
2 seconds	Raw	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)
	Deepsense	79.32 (1.33)	73.05 (1.54)	79.32 (1.33)
5 seconds	Raw	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)
	Deepsense	N/A	N/A	N/A

TABLE II

COMPARISON IN MODEL SIZE AND COMPUTATIONAL COST.

Time	Network	Parameter	MAdds
2 seconds	Raw	0.99M	30M
	Frequency	0.99M	50M
	RawFrequency	1.8M	120M
	Deepsense	68.6M	330M
5 seconds	Raw	2.17M	70M
	Frequency	2.17M	190M
	RawFrequency	4.16M	310M
	Deepsense	427.1M	840M

TABLE III

EER (%) WITH DIFFERENT NUMBER OF FEATURES.

Feature	Mean	SD	Minimum	Median	Maximum
15	3.3	1.8	0.6	3.1	9.4
20	2.6	1.2	0.2	2.5	5.9
25	2.3	1.2	0	2.3	6.1
30	2.5	1.3	0	3.1	7.8
50	3.3	1.6	0	3.1	7.8
100	8.5	3.7	1.2	8.8	17.2

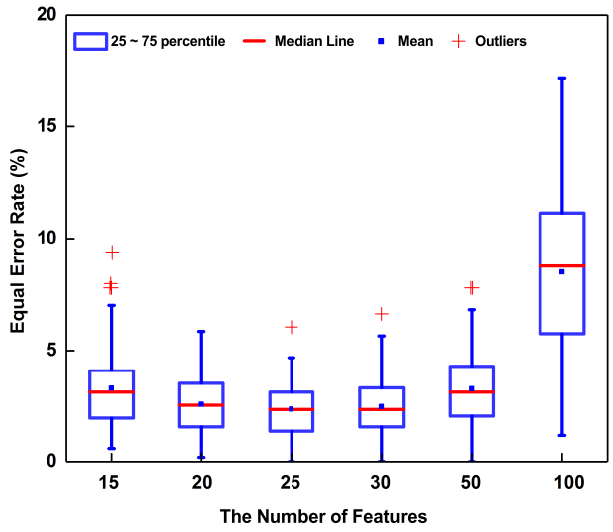

Fig. 4. EER with different number of features.

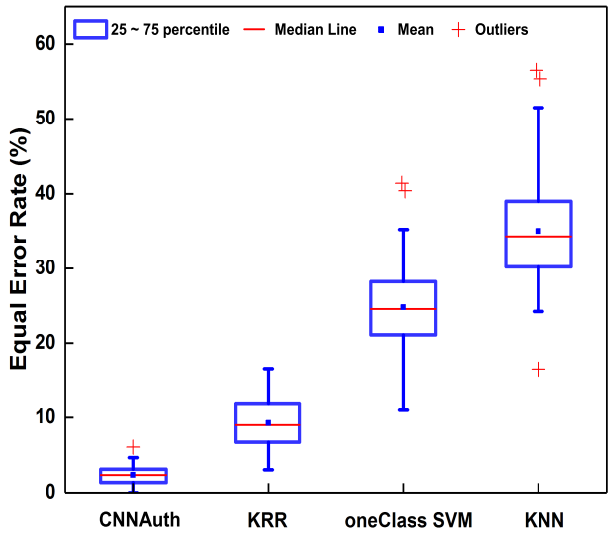

Fig. 5. Comparison with other algorithms.

1) EER: Fig. 4 shows the box plot of EERs with different number of features. These features are selected by PCA. As shown in Fig. 4, the EER decreases with the increase of the feature number, then increases, and finally reaches an optimal feature number at 25. Table III lists the mean, standard deviation (SD), minimum, median and maximum of the EER with different number of features. Therefore, we choose 25 features in *CNNAuth*, which achieves a mean EER of 2.3%.

2) Time efficiency: The time cost of *CNNAuth* 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 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, respectively, for one sample ($t_2 = 169ms$, $t_3 = 1ms$). 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 *CNNAuth*.

3) Comparison with other authentication algorithms: In order to further verify the effectiveness of *CNNAuth*, we compare *CNNAuth* with other common algorithms based on manually designed features.

Fig. 5 shows our approach *CNNAuth* surpasses other classifiers with a large margin of 7.27% at least. In addition, the performance of KRR is the best among algorithms of one-class

SVM and kNN, approximately reaching an EER of 9.26%.

VII. Conclusion

In this paper, we propose a novel two-stream convolutional neural network based authentication system, *CNNAuth*, for continuously authenticating users leveraging their behavioral patterns. *CNNAuth* consists of data collection, data preprocessing, feature extraction, feature selection, classifier, and authentication. First, we design a two-stream CNN to learn and extract the effective and efficient data feature representations, and the most discriminable ones are further selected by the PCA. Then, we use the one-class SVM to train the classifier in the enrollment phase. With the trained classifier and testing features, *CNNAuth* 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 *CNNAuth* with respect to the EER and time efficiency, respectively. The experimental results show that the two-stream CNN achieves the best performance with accuracy of 87.14%, macro F1 of 82.99%, micro F1 of 87.14%, model parameters of 1.8M, and computational cost of 120M, and *CNNAuth* reaches the lowest authentication EER of 2.3% and consumes approximately 3 seconds for authentication, respectively.

References

[1] U. Mahbub, V. M. Patel, D. Chandra, B. Barbello, and R. Chellappa, "Partial face detection for continuous authentication," *in Proc. IEEE International Conference on Image Processing (ICIP)*, pp. 2991–2995, 2016.

[2] A. Hadid, J. Y. Heikkila, O. Silven, and M. Pietikainen, "Face and eye detection for person authentication in mobile phones," *in Proc. ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 101–108, 2007.

[3] N. Shabrina, T. Isshiki, and H. Kunieda, "Fingerprint authentication on touch sensor using phase-only correlation method," *in Proc. 7th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, pp. 85–89, 2016.

[4] L. Hong and A. K. Jain, "Integrating faces and fingerprints for personal identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1295–1307, 1998.

[5] C. Sanchezavila and R. Sanchezreillo, "Two different approaches for iris recognition using gabor filters and multiscale zero-crossing representation," *Pattern Recognition*, vol. 38, no. 2, pp. 231–240, 2005.

[6] I. Martinovic, K. B. Rasmussen, M. Roeschlin, and G. Tsudik, "Authentication using pulse-response biometrics," *Communications of The ACM*, vol. 60, no. 2, pp. 108–115, 2017.

[7] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi, "Tips: context-aware implicit user identification using touch screen in uncontrolled environments," *in Proc. 15th Workshop on Mobile Computing Systems and Applications (HotMobile)*, pp. 1–6, 2014.

[8] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," *in Proc. 28th International Conference on Neural Information Processing Systems (NIPS)*, pp. 2449–2457, 2015.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *in Proc. 25th International Conference on Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *in Proc. International Conference on Learning Representations (ICLR)*, pp. 1–14, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *in Porc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

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

[13] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *in Porc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9.

[14] M. B. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *in Porc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

[15] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli, "On the use of sift features for face authentication," *in Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, pp. 1–7, 2006.

[16] H. Feng, K. Fawaz, and K. G. Shin, "Continuous authentication for voice assistants," *in Proc. 23rd ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pp. 343–355, 2017.

[17] B. Zou and Y. Li, "Touch-based smartphone authentication using import vector domain description," *in Proc. 29th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 85–88, 2018.

[18] Y. Li, H. Hu, G. Zhou, and S. Deng, "Sensor-based continuous authentication using cost-effective kernel ridge regression," *IEEE Access*, vol. 6, pp. 32554–32565, 2018.

[19] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *in Proc.27th International Conference on Neural Information Processing Systems (NIPS)*, pp. 568–576, 2014.

[20] L. Sifre and S. Mallat, "Rigid-motion scattering for texture classification," *International Journal of Computer Vision*, vol. 3559, pp. 501–515, 2014.

[21] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *in Porc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

[22] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," *in Proc. 20th Annual Network and Distributed System Security Symposium (NDSS)*, pp. 1–16, 2013.

[23] C. Bo, L. Zhang, X. Li, Q. Huang, and Y. Wang, "Silentsense: silent user identification via touch and movement behavioral biometrics," *in Proc. 19th Annual International conference on Mobile Computing and Networking (MobiCom)*, pp. 187–190, 2013.

[24] Y. Li, H. Hu, and G. Zhou, "Using data augmentation in continuous authentication on smartphones," *IEEE Internet of Things Journal*, pp. 1–13, 2018. DOI: 10.1109/JIOT.2018.2851185.

[25] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of Machine Learning Research*, vol. 2, no. 2, pp. 139–154, 2002.

[26] Z. Sitova, J. Sedenka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani, "Hmog: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 877–892, 2016.

[27] W. Lee and R. B. Lee, "Implicit sensor-based authentication of smartphone users with smartwatch," *in Proc. the Hardware and Architectural Support for Security and Privacy (HASP)*, pp. 1–8, 2016.

[28] S. Buthpitiya, Y. Zhang, A. K. Dey, and M. L. Griss, "n-gram geo-trace modeling," *in Porc. International Conference on Pervasive Computing (ICPC)*, pp. 97–114, 2011.

[29] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *in Proc. 26th International Conference on World Wide Web (WWW)*, pp. 351–360, 2017.

[30] 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 Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 404–416, 2017.

[31] C. Shen, T. Yu, S. Yuan, Y. Li, and X. Guan, "Performance analysis of motion-sensor behavior for user authentication on smartphones," *Sensors*, vol. 16, no. 3, p. 345, 2016.

[32] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *in Proc. International Conference on Learning Representations (ICLR)*, pp. 1–15, 2015.