

## Singapore Management University Institutional Knowledge at Singapore Management University

---

Research Collection School Of Information Systems

School of Information Systems

---

5-2018

# Breathing-based authentication on resource-constrained IoT devices using recurrent neural networks

Jagmohan CHAUHAN

Suranga SENEVIRATNE

Yining HU

Archan MISRA

Singapore Management University, [archanm@smu.edu.sg](mailto:archanm@smu.edu.sg)

Aruna SENEVIRATNE

*See next page for additional authors*

**DOI:** <https://doi.org/10.1109/MC.2018.2381119>

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)

 Part of the [Digital Communications and Networking Commons](#), and the [OS and Networks Commons](#)

---

### Citation

CHAUHAN, Jagmohan; SENEVIRATNE, Suranga; HU, Yining; MISRA, Archan; SENEVIRATNE, Aruna; and LEE, Youngki. Breathing-based authentication on resource-constrained IoT devices using recurrent neural networks. (2018). *Computer*. 51, (5), 60-67. Research Collection School Of Information Systems.

**Available at:** [https://ink.library.smu.edu.sg/sis\\_research/4054](https://ink.library.smu.edu.sg/sis_research/4054)

This Journal Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [libIR@smu.edu.sg](mailto:libIR@smu.edu.sg).

---

**Author**

Jagmohan CHAUHAN, Suranga SENEVIRATNE, Yining HU, Archan MISRA, Aruna SENEVIRATNE, and Youngki LEE



# Breathing-Based Authentication on Resource-Constrained IoT Devices using Recurrent Neural Networks

**Jagmohan Chauhan**, Aalto University

**Suranga Seneviratne**, University of Sydney

**Yining Hu**, Data61 and University of New South Wales

**Archan Misra**, Singapore Management University

**Aruna Seneviratne**, University of New South Wales

**Youngki Lee**, Singapore Management University

*Recurrent neural networks (RNNs) have shown promising results in audio and speech-processing applications. The increasing popularity of Internet of Things (IoT) devices makes a strong case for implementing RNN-based inferences for applications such as acoustics-based authentication and voice commands for smart homes. However, the feasibility and performance of these inferences on resource-constrained devices remain largely unexplored. The authors compare traditional machine-learning models with deep-learning RNN models for an end-to-end authentication system based on breathing acoustics.*

**S**ensors embedded in smartphones, wearables, and other Internet of Things (IoT) devices are increasingly being used to support both fine-grained monitoring of users' activities/ambient context and richer forms of cyber-physical interaction (via gestures or natural language interfaces). Illustrative scenarios include monitoring users' steps to estimate daily calorie expenditure, tracking eating gestures to capture food intake, and the use of microphone-equipped devices for voice-based home-automation control. Because these personal and edge devices learn and store individual-specific information and increasingly come in nontraditional forms, it is important to develop secure and usable user authentication techniques.

In a recent work, we introduced the BreathPrint system, which utilizes acoustic features of a user's breathing (captured by a commodity microphone) to support ubiquitous user authentication on mobile and IoT devices.<sup>1</sup> In this article, we investigate whether recurrent neural network (RNN)-based deep-learning models can be effectively used on resource-constrained devices for BreathPrint. The original work on BreathPrint was cloud-based and used a conventional Gaussian mixture model (GMM)-based machine-learning model with manually curated features.

This work is motivated by two main objectives. First, state-of-the-art speech recognition and speaker identification methods are RNN-based and significantly outperform classical methods such as support vector machines (SVMs), GMMs, and hidden Markov models (HMMs), especially in noisy environments,<sup>2</sup> so it is important to evaluate the performance of BreathPrint using RNN models to boost the performance. Secondly, for

unobtrusive and ubiquitous authentication applications, BreathPrint needs to be implemented on resource-constrained devices and must be able to authenticate users without cloud access. Thus, it is important to obtain an idea of the accuracy-resource tradeoff for BreathPrint.

To this end, we conducted a performance evaluation of an end-to-end RNN-based BreathPrint authentication system on three representative hardware platforms: mobile (smartphone), wearable (smartwatch), and IoT (Raspberry Pi 3). To the best of our knowledge, this is the first work that shows the feasibility and performance of RNN-based deep-learning models for acoustics on limited-resource footprint devices.

We make the following key contributions.

- › We present performance evaluation results of an end-to-end authentication system based on both shallow models (SVM) and long short-term memory (LSTM, a variant of RNN) using breathing acoustics on three representative IoT devices: smartphone, smartwatch, and Raspberry Pi.
- › We show that RNN-based models for acoustic classification are smaller in size and more lightweight than previously reported results with convolutional neural network (CNN)-based models, and thus can be adopted on IoT devices. Specifically, an uncompressed RNN model is only 1.1 Mbytes in size (for relevant breathing gestures) and can run on smartphones and smartwatches with a latency of approximately 100 to 200 ms and 700 to 1,000 ms, respectively.

- › We show how a linear quantization-based model-compression technique can help reduce the memory footprint of RNN models by a factor of 5 (to approximately 150 to 50 Kbytes) without suffering any consequential drop in accuracy.

## RELATED WORK

Motivated by the breakthroughs in training deep neural networks (DNNs) and the impressive performance gains (deep networks have outperformed conventional machine-learning models and even human experts), a variety of recent work has focused on the challenges (such as higher memory requirement or excessive computational latency) of executing deep-learning models on resource-constrained devices. Table 1 summarizes some of these notable efforts and the techniques employed. Broadly speaking, these efforts utilize one or more of the following three approaches: offloading neural network processing to GPUs, which are more efficient in vectorized computations; reducing the time and memory requirements to load the fully connected layers; and faster execution of the convolutional layer tasks.

Early work by Nicholas D. Lane and his colleagues investigated the performance characteristics, resource requirements, and execution bottlenecks for deep-learning models (CNN and DNN) on mobile, wearable, and IoT devices to support audio- and vision-based apps.<sup>7</sup> Results indicated that although smaller deep-learning models work without issues on these devices, more complex CNN models such as AlexNet do not work well under the resource constraints. To address this problem, Sourav Bhattacharya

**TABLE 1.** Deep learning (DL) on resource-constrained devices.

Name	Type of DL	Architecture	Application	Techniques
SparseSep <sup>3</sup>	Convolutional neural network (CNN), deep neural network (DNN)	Multiple layers	Image classification, speaker identification, and scene analysis	Sparsification and separation
DeepEar <sup>2</sup>	DNN	5 layers	Audio sensing	NA
DeepEye <sup>4</sup>	CNN	Multiple layers	Continuous vision	Interleaving, caching, and compression
DeepMon <sup>5</sup>	CNN	16 layers	Continuous vision	GPU offloading, caching, and decomposition
MobiRNN <sup>6</sup>	Recurrent neural network (RNN)	2 layers	Activity recognition	GPU offloading

and his colleagues proposed Spars-eSep, which focuses primarily on finding a sparse representation of the fully connected layers and using separate filters for the convolutional kernels.<sup>3</sup> These techniques reduce the number of parameters and convolutional operations required to execute a deep-learning model and can thus significantly reduce the computational and space complexity on resource-constrained devices.

Several works have focused on optimizing deep networks for audio- and image-sensing applications. DeepEar is an audio-sensing application for smartphones based on DNNs.<sup>2</sup> It was implemented in the digital signal processor of a smartphone and imposed only 6 percent additional overhead in daily energy consumption. DeepEye deploys CNNs on wearables for continuous vision applications.<sup>4</sup> It avoids resource bottlenecks by using interleaving to orchestrate the execution of computation-heavy convolutional layers with memory-heavy fully connected layers. DeepEye also employs caching to load the fully connected layers faster and utilizes a singular value decomposition (SVD)-based layer-factorization approach to compress the fully connected layer. DeepMon focuses on reducing the processing latency of convolutional layers (via multiple optimization techniques) for continuous vision applications.<sup>5</sup> It uses a combination of caching (exploiting similarities between successive images) and model decomposition (breaking up

convolutional layers into smaller layers) to reduce the computational overhead. DeepMon also offloads tasks in convolutional layers to mobile GPUs for faster processing. Finally, MobiRNN applies GPU offloading to execute RNNs faster on smartphones to support activity-recognition tasks.<sup>6</sup>

As evident from Table 1, most of the work to date has focused on CNNs and DNNs. For example, even audio analysis and speaker identification tasks have been performed using CNN and DNNs. In general, CNNs are good at exploiting features defined on spatial data (such as images), whereas RNNs are more appropriate for identifying and using temporal features defined over datastreams (such as audio or text). RNN-based models are less complex than CNN-based models as they deal with less complex data and do not use convolutional filters. They also require lower computational power and memory than CNN-based models.<sup>6,7</sup>

BreathPrint proposes a technique to authenticate users based on their breathing acoustics on mobile and IoT devices. It is based on the hypothesis that each individual's breathing pattern is unique. The proposed approach is also highly usable, as it merely requires the user to perform a few breathing gestures. Our interest in RNNs is thus driven by our belief that this uniqueness is manifested via temporal variations in a user's breathing pattern, and that RNNs are more capable of identifying and exploiting such temporal features.

However, BreathPrint's full potential can only be realized if the user identification can be performed locally (on the device) and with minimal latency. Accordingly, in this work, we investigate the central question: "Can BreathPrint be practically realized using an RNN-based model on resource-constrained devices?"

Figure 1 shows some real-world applications where BreathPrint can be used as an entry point or a continuous authentication mechanism. As an entry point authentication, the primary use case is unlocking smart devices such as smartphones and wearables. Unlocking devices is required to get access to the device as well as to get personalized user experiences. For example, in an industrial setting, an augmented-reality smartglass that overlays product-specific knowledge on equipment can adjust content and instructions based on the individual worker's profile. Other use cases include operating IoT devices such as sensor-enabled doors or coffee makers in smart spaces. The user needs to re-authenticate if the device goes to sleep or is in inactive mode. In addition, BreathPrint can be employed on smart respirators to adjust humidity levels to match an individual worker's preferences in an industrial setting, or it can continuously authenticate users based on their breathing patterns. The re-authentication time in a continuous authentication scheme can be configured according to the required levels of security. Continuous authentication

does not require user intervention unless re-authentication fails.

This work differs significantly from BreathPrint in numerous ways. First, the two works have different focuses. BreathPrint introduces a new behavioral modality for user authentication. The focus of this work is to evaluate the performance of shallow and deep-learning models to see if BreathPrint can be used to perform real-time authentication on resource-constrained devices. Secondly, this work deals with user identification compared to BreathPrint's user verification. In user identification, the job of an authentication system is to predict which user is trying to get access from a closed set of users. In user verification, an authentication system checks if a user X trying to access the system is actually user X.

## EXPERIMENTAL SETUP

To evaluate the feasibility of RNN-driven breathing-based authentication, we utilized the breathing acoustics dataset collected in our previous work.<sup>1</sup> The dataset consists of acoustic samples of three breathing gestures—deep breathing, normal breathing, and sniffing (two quick inhalations)—of 10 users collected over three sessions. For each gesture, the dataset contains 30, 30, and 10 samples collected on the first day (session 1), the fourth day (session 2), and the seventh or eighth day (session 3), respectively. In this article, we focus only on two breathing gestures (deep breathing and sniffing), as our earlier investigations revealed that these two perform better in authentication applications compared to normal breathing. Refer to our original article for further details on the dataset.<sup>1</sup>

For each user, we selected the first 50 samples for model training and tuning

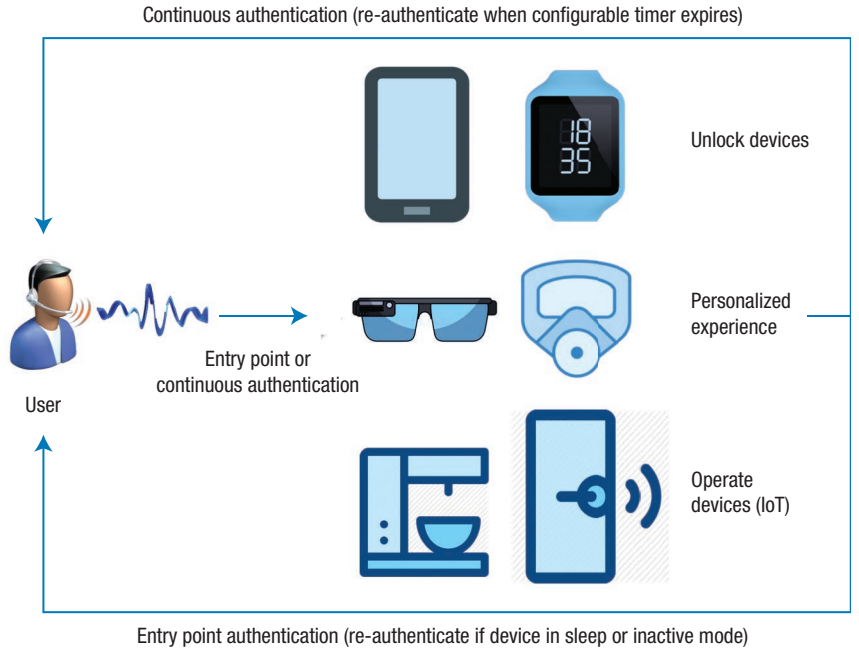


FIGURE 1. Applications of BreathPrint.

TABLE 2. Hardware configuration of the devices..

Device	OS	CPU	GPU	Memory
Nexus 5 smartphone	Android 6.0	2.26 GHz Quadcore	Adreno 330	2 Gbytes
Pixel smartphone	Android 7.0	2.15 and 1.6 GHz Quadcore	Adreno 530	4 Gbytes
LG G Watch R	Android Wear 2.0	1.2 GHz Quadcore	Adreno 305	512 Mbytes
Raspberry Pi 3	Android Things 5.0	1.2 GHz Quadcore	VideoCore IV	1 Gbyte

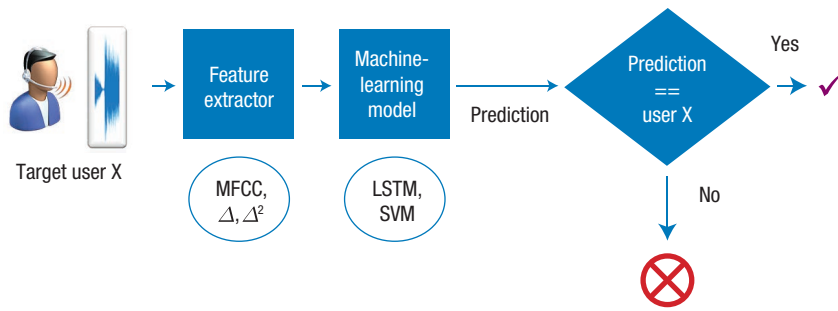
purposes. Because a deep-learning model requires larger sample sizes for training, we applied two commonly used data-augmentation techniques to increase the number of data samples. We used a combination of frequency wrapping<sup>8</sup> and amplitude scaling.<sup>9</sup> Each sample was scaled 10 times along the time axis and the amplitude by selecting two separate values from a uniform distribution;  $\sim U(0.8, 1.2)$ . Overall, we obtained an 11-fold boost in the number of training examples (550 training samples per participant), consisting of both original samples and their augmented versions. The remaining 10 original samples (from sessions 2 and 3) were kept intact for testing.

We performed an experimental evaluation using four devices belonging to three distinct types (listed in Table 2)—two smartphones (mobile), a smartwatch (wearable), and a Raspberry Pi (IoT). All the devices run a different variant of an Android-based OS and are representative of popular commercial mobile, wearable, and embedded platforms.

## METHODOLOGY

Our overall goal is user authentication, or deciding whether a sample belongs to one of  $N$  preregistered possible users. This is effectively a problem of closed-set user identification that can be mapped as a multi-class classification problem (where a

## EMBEDDED DEEP LEARNING



**FIGURE 2.** User identification. MFCC: mel-frequency cepstral coefficients; LSTM: long short-term memory; SVM: support vector machine.

single class represents one user). Figure 2 shows our system. A target user X first does a breathing gesture on the microphone of a device (for example, to unlock the device) from which features are extracted. The features are then put into a classifier to get the prediction output. If the predicted user is the same as the target user, then the authentication is successful.

Our approach involved some preprocessing of the original acoustics signal so that it could be fed into an RNN model. For a comparative baseline that uses a shallow classifier, we also trained an SVM model.

### Feature extraction

We divided each audio file into 10-ms nonoverlapping frames with Hamming window-based smoothing. For each frame, we calculated 96 mel-frequency cepstral coefficients (MFCC) features (32 MFCC, 32 Delta MFCC, and 32 Double Delta MFCC) using JSTK (Java Speech Toolkit; <https://github.com/sikoried/jstk>). Then we used windowing to combine these frames so that temporal information between the frames is retained. For sniffing and deep-breathing gestures, we tried window sizes of length {20, 25, 30, 35} and {200, 250, 300, 350}, respectively. There are two factors to consider when selecting a suitable window size. Each window must be large enough to retain a significant part of a breathing gesture. However, the duration of a single breath varies significantly across users; consequently, if a larger window size is selected, some testing samples could be missed from users

with relatively short breathing duration. To balance these considerations, we ended up with window sizes {20, 25, 30} for sniffing and {200, 250} for deep-breathing gestures, respectively. To further augment the training dataset, we created overlapping windows for a given breathing sample. We chose three overlap sizes (90 percent, 70 percent, and 50 percent) of the window size. For each window size and overlap value pair, we trained the classifiers as discussed below.

### Training and testing datasets

We used the samples created from augmentation for the training. More specifically, we created overlapping windows from these samples and randomly shuffled them—we used 80 percent of the windows as the training set and the rest (20 percent) as the validation set to tune hyperparameters. We refer to the overlapping windows created from the rest of the 10 audio samples in session 2 as the intra-set and overlapping windows created from the 10 audio samples in session 3 as the inter-set.

### Models

We used an RNN architecture similar to the one described in Nils Y. Hammerla and his colleagues' work.<sup>10</sup> This architecture is illustrated in Figure 3. The hidden unit size of the LSTM units was 128, and we used two LSTM layers. We implemented the model using Tensorflow. We used 32 as the batch size and trained the network over 500 iterations. As a baseline classifier, we also trained a multiclass SVM classifier with a linear kernel using

LIBSVM ([www.csie.ntu.edu.tw/~cjlin/libsvm/](http://www.csie.ntu.edu.tw/~cjlin/libsvm/)). Testing LIBSVM on Android devices utilizes LIBSVM-AndroidJNI (<https://github.com/cnbuff410/libsvm-androidjni>). Note that while an SVM model with a nonlinear kernel might provide better results than a linear kernel, we found that because of the large number of support vectors, SVM models with a nonlinear kernel could not be loaded on any of the devices. The number of support vectors included in an SVM model with a nonlinear kernel is higher than the support vectors needed for a linear kernel. GMM is not suitable for solving multiclass classification problems due to scalability issues. SVM is more suitable for multiclass classification. Because of the issues present with nonlinear kernels and GMM, we evaluated SVM with a linear kernel as a baseline.

### Model selection

We used early termination to select the best model because we observed that after some iterations, the model accuracy reached the maximum and stayed approximately in the same region for the validation set, while the accuracy in the intra- and inter-sets showed a slight declining trend.

To select a suitable elbow point of the accuracy graphs, we first applied 20-point moving averaging to the validation set accuracy graph and then selected the point from which the accuracy does not improve by 5 percent for the next four consecutive points. An accuracy graph shows the accuracy achieved at different iterations for a deep-learning model. The moving average window and the improvement threshold was decided empirically. Once the elbow point was decided, we selected 11 models (five previous models and the five next

models including the model at the elbow point). To present the performance results, we selected the models that gave the highest average accuracy over all three datasets. The window and overlap configurations that gave the highest accuracy were window size 30 with 90 percent overlap for sniffing and window size 250 with 90 percent overlap for deep breathing. For SVM, we picked the model with the best cross-validation accuracy.

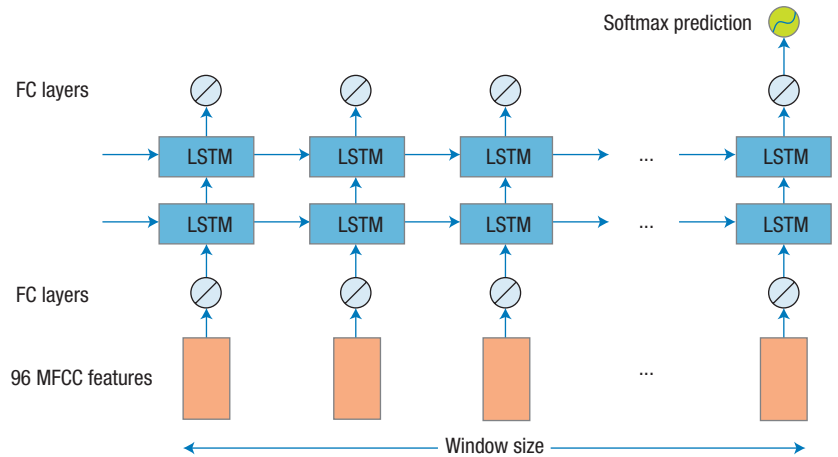
### Model reduction

To empirically study the computational overhead versus the latency-accuracy tradeoff, we compressed the selected RNN models using the inbuilt 256-level quantization function provided in Tensorflow ([www.tensorflow.org/performance/quantization](http://www.tensorflow.org/performance/quantization)). This function extracts the minimum and maximum for each layer and then compresses each float value to an eight-bit integer. Note that this option will save space in zipped formats that are usually used in Android applications. We executed the original as well as quantized models.

## RESULTS

We used four performance metrics for our experimental evaluation.

1. Accuracy: the percentage of correct user identifications.
2. Feature extraction time: time taken to extract MFCC features from an audio file.
3. Model loading time: time to load the machine-learning model into the memory.
4. Inference time: time to predict the user label once the feature extraction is done and the learning model is loaded into memory.



**FIGURE 3.** Recurrent neural network (RNN) architecture. FC: fully connected.

### Performance of LSTM

Here, we report average values of the execution times for different phases of the classification process. The feature extraction time was in the range of 40 to 60 ms for the sniffing gesture and in the range of 126 to 484 ms for the deep-breathing gesture across the four devices. As expected, the Pixel smartphone (the most powerful platform with the highest memory) and the smartwatch impose the least and highest feature extraction times, respectively. Figure 4 plots the results for the other three metrics. Note that the time scale on the y-axis is in log scale.

We made the following observations.

- › Model loading time decreases linearly with the RAM of the device. Loading the LSTM model to the Pixel smartphone takes around 100 ms (4 Gbytes RAM), while the same model takes roughly twice the time (200 ms) on the Nexus 5 smartphone (2 Gbytes RAM), four times longer (400 ms) on Raspberry Pi (1 Gbyte RAM), and seven times longer (700 ms) on the smartwatch (512 Mbytes RAM) for the sniffing gesture. The same observation holds for the deep-breathing gesture.
- › Inference time depends on the processing power of the device. The inference time (using the LSTM model) for the sniffing gesture is on average 23 ms for

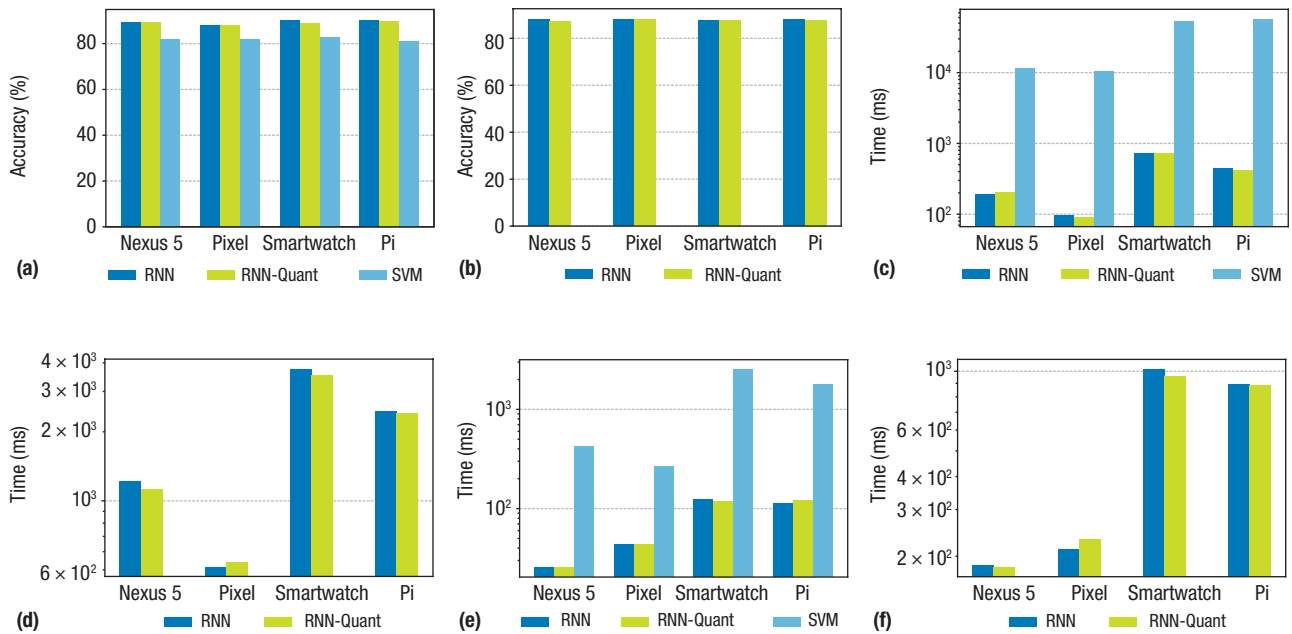
Nexus 5, 40 ms for Pixel, and around 100 ms for both the smartwatch and Raspberry Pi. In contrast, the inference time for the deep-breathing gesture is approximately 180 ms for Nexus 5, 200 ms for Pixel, 900 ms for Raspberry Pi, and 1,000 ms for the smartwatch.

- › The accuracy of the system was around 90 percent for both gestures for the intra-set. We also observed that the accuracy of LSTM models was slightly higher than SVM. This indicates that the deep-learning model can perform at least as well as the alternative SVM approach, even though the volume of training data was fairly small. Note that the accuracy of user identification drops to 75 percent and 70 percent for the sniffing and deep-breathing gestures, respectively, when applied to the inter-set data. This result is consistent with our prior results<sup>1</sup> and suggests that a larger training corpus will be needed to accommodate a wider range of context-dependent variations in breathing patterns.

### Benefit of quantization

The results in Figure 4a and 4b also show that the use of a quantized model does not result in any loss of classification accuracy (compared to the





**FIGURE 4.** Metrics for different breathing gestures: accuracy–sniff (a), accuracy–deep (b), model loading time–sniff (c), model loading time–deep (d), inference time–sniff (e), inference time–deep (f). RNN-Quant refers to the quantized model.

original LSTM model). However, quantization reduces the size of the model and offers some execution benefits. The size of the quantized model is 175 Kbytes (compared to 1.1 Mbytes for the uncompressed model) for the sniffing gesture and 264 Kbytes (compared to 1.1 Mbytes for the uncompressed model) for the deep-breathing gesture. The quantized model thus reduces the memory footprint significantly (by a factor of 4 to 6), enabling it to be loaded a bit faster.


### LSTM versus SVM

Contrary to popular belief, the shallow SVM model does provide accuracy comparable to the LSTM model, but takes much longer (50 to 100 seconds) to load into memory. Moreover, the execution time taken to predict the user label is also 5 to 20 times higher than the corresponding time for the LSTM-based model across all devices. The best model we tested with SVM for the sniffing gesture is 280 Mbytes in size. Comparatively, the LSTM model size is 2 Mbytes when uncompressed and only a few hundred Kbytes when quantized. The large size of SVM models is due to the large number of

support vectors needed to support the multiclass classification, which does not happen for binary class classification. In fact, the size of the SVM model for deep breathing is 2 Gbytes or higher, so it could not be loaded on any of the devices. Our results suggest that the LSTM-based deep-learning model is more lightweight and robust (especially with the quantized LSTM model) than the SVM-based shallow classifier.

Our experiments reveal that an RNN-based approach for user authentication based on breathing acoustics is not only robust but is also lightweight enough to be effectively executed on a variety of resource-constrained embedded devices. An appropriately quantized LSTM-based deep-learning model can authenticate users with accuracy higher than 90 percent and utilizes models that are modestly sized (a couple of hundred Kbytes). The resulting user authentication latency is small not only for representative smartphones (less than or equal to 200 ms) but also for the highly resource-limited smartwatch platform (less than or equal

to 1 second). Note that these performance numbers are achieved using CPU-only computation and should be significantly improved using GPU-offloading approaches proposed by other researchers.

Our investigations suggest that RNNs offer a compelling lightweight alternative to CNNs for many sensor-driven pervasive applications, especially if the application utilizes temporal features of the underlying sensor data. 

### REFERENCES

1. J. Chauhan et al., “BreathPrint: Breathing Acoustics-Based User Authentication,” *Proc. 15th Ann. Int’l Conf. Mobile Systems, Applications, and Services (MobiSys 17)*, 2017, pp. 278–291.
2. N.D. Lane, P. Georgiev, and L. Qendro, “DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments Using Deep Learning,” *Proc. 2015 ACM Int’l Joint Conf. Pervasive and Ubiquitous Computing (UbiComp 15)*, 2015, pp. 283–294.
3. S. Bhattacharya and N.D. Lane, “Sparsification and Separation of Deep Learning Layers for

## ABOUT THE AUTHORS

**JAGMOHAN CHAUHAN** is a visiting researcher at Aalto University. His research interests include performance and security of mobile and IoT systems and deep learning. Chauhan received a PhD in electrical engineering and telecommunications from the University of New South Wales (UNSW). Contact him at [jagmohan.chauhan@aalto.fi](mailto:jagmohan.chauhan@aalto.fi).

**SURANGA SENEVIRATNE** is a lecturer in security at the University of Sydney's School of IT. His research interests include privacy and security in mobile systems, AI applications in security, and behavioral biometrics. Seneviratne received a PhD in electrical engineering and telecommunications from UNSW. Contact him at [suranga.seneviratne@sydney.edu.au](mailto:suranga.seneviratne@sydney.edu.au).

**YINING HU** is a research assistant at CSIRO's Data61, where she mainly works on wearable-related topics. Her research interests include designing blockchain-based micropayment systems in areas with intermittent network connectivity. Hu received a bachelor's degree in electrical engineering from the University of Sydney and the Harbin Institute of Technology (joint program). She is pursuing her PhD in electrical engineering and telecommunications at UNSW. Contact her at [yining.hu@data61.csiro.au](mailto:yining.hu@data61.csiro.au).

**ARCHAN MISRA** is a professor and the associate dean of research in the School of Information Systems at Singapore Management University (SMU). His research interests include novel wearable and IoT sensing and analytics techniques, and the use of such techniques to build context-aware mobile systems in domains such as retail, smart manufacturing, and smart homes. Archan has worked extensively on problems spanning wireless networking, mobile and pervasive computing, and urban sensing. He received a PhD in electrical and computer engineering from the University of Maryland at College Park, and he chaired the IEEE Computer Society's Technical Committee on Computer Communications (TCCC) from 2005 to 2007. Contact him at [archanm@smu.edu.sg](mailto:archanm@smu.edu.sg).

**ARUNA SENEVIRATNE** is a professor at UNSW, where he holds the Mahanakorn Chair of Telecommunications. His research interests include mobile technologies, networking and communications, and computer system security. Seneviratne received a PhD in electrical engineering from the University of Bath. Contact him at [aruna.seneviratne@data61.csiro.au](mailto:aruna.seneviratne@data61.csiro.au).

**YOUNGKI LEE** is an assistant professor at SMU. His research interests include building underlying mobile and sensor platforms to enable always available and highly enriched awareness of human behavior and contexts, and building innovative life-immersive mobile applications in various domains such as daily wellbeing and childcare. Lee received a PhD in computer science from KAIST. Contact him at [youngkilee@smu.edu.sg](mailto:youngkilee@smu.edu.sg).

- Constrained Resource Inference on Wearables," *Proc. 14th ACM Conf. Embedded Network Sensor Systems (SenSys 16)*, 2016, pp. 176–189.
4. A. Mathur et al., "DeepEye: Resource Efficient Local Execution of Multiple Deep Vision Models Using Wearable Commodity Hardware," *Proc. 15th Ann. Int'l Conf. Mobile Systems, Applications, and Services (MobiSys 17)*, 2017, pp. 68–81.
  5. L.N. Huynh, Y. Lee, and R.K. Balan, "DeepMon: Mobile GPU-Based Deep Learning Framework for Continuous Vision Applications," *Proc. 15th Ann. Int'l Conf. Mobile Systems, Applications, and Services (MobiSys 17)*, 2017, pp. 82–95.
  6. Q. Cao, N. Balasubramanian, and A. Balasubramanian, "MobiRNN: Efficient Recurrent Neural Network Execution on Mobile GPU," *Proc. 1st Int'l Workshop Deep Learning for Mobile Systems and Applications (EMDL 17)*, 2017, pp. 1–6.
  7. N.D. Lane et al., "An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices," *Proc. 2015 Int'l Workshop Internet of Things towards Applications (IoT-App 15)*, 2015, pp. 7–12.
  8. M.T. Islam, B. Islam, and S. Nirjon, "SoundSifter: Mitigating Overhearing of Continuous Listening Devices," *Proc. 15th Ann. Int'l Conf. Mobile Systems, Applications, and Services (MobiSys 17)*, 2017, pp. 29–41.
  9. D.T. Nguyen et al., "SwallowNet: Recurrent Neural Network Detects and Characterizes Eating Patterns," *IEEE Int'l Conf. Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2017, pp. 401–406.
  10. N.Y. Hammerla, S. Halloran, and T. Plotz, "Deep, Convolutional, and Recurrent Models for Human

Activity Recognition Using Wearables," *Proc. 25th Int'l Joint Conf. Artificial Intelligence (IJCAI 16)*, 2016, pp. 1533–1540.