

Performance Characterization of Deep Learning Models for Breathing-based Authentication on Resource-Constrained Devices

JAGMOHAN CHAUHAN*, Aalto University, Finland
JATHUSHAN RAJASEGARAN, University of Moratuwa, Sri Lanka
SURANGA SENEVIRATNE, University of Sydney, Australia
ARCHAN MISRA, Singapore Management University, Singapore
ARUNA SENEVIRATNE, University of New South Wales, Australia
YOUNGKI LEE, Seoul National University, South Korea

Providing secure access to smart devices such as smartphones, wearables and various other IoT devices is becoming increasingly important, especially as these devices store a range of sensitive personal information. Breathing acoustics-based authentication offers a highly usable and possibly a secondary authentication mechanism for secure access. Executing sophisticated machine learning pipelines for such authentication on such devices remains an open problem, given their resource limitations in terms of storage, memory and computational power. To investigate this challenge, we compare the performance of an end-to-end system for both user identification and user verification tasks based on breathing acoustics on three type of smart devices: smartphone, smartwatch and Raspberry Pi using both shallow classifiers (i.e., SVM, GMM, Logistic Regression) and deep learning based classifiers (e.g., LSTM, MLP). Via detailed analysis, we conclude that LSTM models for acoustic classification are the smallest in size, have the lowest inference time and are more accurate than all other compared classifiers. An uncompressed LSTM model provides an average f-score of 80%-94% while requiring only 50–180 KB of storage (depending on the breathing gesture). The resulting inference can be done on smartphones and smartwatches within approximately 7–10 ms and 18–66 ms respectively, thereby making them suitable for resource-constrained devices. Further memory and computational savings can be achieved using model compression methods such as weight quantization and fully connected layer factorization: in particular, a combination of quantization and factorization achieves 25%–55% reduction in LSTM model size, with almost no loss in performance. We also compare the performance on GPUs and show that the use of GPU can reduce the inference time of LSTM models by a factor of 300%. These results provide a practical way to deploy breathing based biometrics, and more broadly LSTM-based classifiers, in future ubiquitous computing applications.

CCS Concepts: • **Computer systems organization** → Embedded and cyber-physical systems; • **Security and Privacy** → Security Services; • **General and reference** → Performance;

Additional Key Words and Phrases: Security, Authentication, Breathing Gestures, Wearables, IoT, SVM, GMM, MLP, LSTM

*This is the corresponding author

Authors' addresses: Jagmohan Chauhan, Aalto University, Espoo, Finland, jagmohan.chauhan@aalto.fi; Jathushan Rajasegaran, University of Moratuwa, Sri Lanka, brjathu@gmail.com; Suranga Seneviratne, University of Sydney, Sydney, NSW, Australia, suranga.seneviratne@sydney.edu.au; Archan Misra, Singapore Management University, Singapore, Singapore, archanm@smu.edu.sg; Aruna Seneviratne, University of New South Wales, Sydney, Australia, aruna.seneviratne@unsw.edu.au; Youngki Lee, Seoul National University, Seoul, South Korea, youngkilee@snu.ac.kr.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Association for Computing Machinery.
2474-9567/2018/12-ART158 \$15.00
<https://doi.org/10.1145/3287036>

Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., Vol. 2, No. 4, Article 158. Publication date: December 2018.

ACM Reference Format:

Jagmohan Chauhan, Jathushan Rajasegaran, Suranga Seneviratne, Archan Misra, Aruna Seneviratne, and Youngki Lee. 2018. Performance Characterization of Deep Learning Models for Breathing-based Authentication on Resource-Constrained Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 158 (December 2018), 24 pages. <https://doi.org/10.1145/3287036>

1 INTRODUCTION

Smartphones are already popular and wearables (especially smartbands, smartglasses and smartwatches) are rapidly gaining market share—some estimates suggest that 504.65 million wearable devices will be sold by the year 2021 [12]. Such pervasive devices are being increasingly used for various sensitive daily applications, such as performing financial transactions [21] or monitoring health and fitness levels [27]. These devices also store other personal information such as user profiles, call records, contacts and emails. Ensuring authorized access to such devices and their services is thus critically important [6, 32]. IoT is another emerging category of devices [34] requiring usable and secure access. Personalized use of wearable and IoT devices will also be an enabler of future smart manufacturing and Industry 4.0 operations. For example, an augmented reality (AR) smartglass, which overlays product-specific knowledge on real-world equipment, can adjust such content and instructions based on the individual worker’s profile (e.g., to tailor to different levels of expertise or different data access privileges).

For an authentication scheme to work on such small form-factor and ubiquitously deployed devices, it has to be not only *secure* but also *unobtrusive*. Also, as many of the wearable and IoT devices are resource-constrained, the authentication mechanism must be *lightweight*. Cloud-based resource augmentation is not always possible and not preferred—e.g., for many IoT devices, continuous Internet connection is not guaranteed. Moreover, many application scenarios require continuous access authorization (e.g., a factory worker using a smartglass should be periodically re-authenticated), suggesting the possible need for lightweight *authentication mechanisms* that augment more traditional techniques. To tackle these goals of *security*, *lightweight local processing* and *unobtrusiveness*, we consider a novel form of authentication, based on *breathing acoustics*, introduced in *BreathPrint* [7]. *BreathPrint* utilizes breathing (a natural human action) as a behavioural fingerprint, with the user being authenticated via acoustics from gestures such as sniff, normal breathing and deep breathing. However, it was deployed as a cloud-based solution and used a shallow machine learning model based on GMM (Gaussian Mixture Model), with handcrafted features.

In this paper, to make *BreathPrint* more *accurate/robust* and *pervasive*, we investigate whether (a) *BreathPrint* can be effectively implemented using deep learning models and (b) whether these deep learning models can be locally executed on resource-constrained devices. We focus on deep learning primarily because they can outperform shallow classifier models based on SVM and GMM/HMM. However, deep learning based classifiers are currently known to impose significant computational overheads, and also require large volumes of training data. To this end, we conducted performance evaluation of an end-to-end LSTM based *BreathPrint* authentication system on three representative hardware platforms: mobile (smartphone), wearable (smartwatch) and IoT (Raspberry Pi). We also investigate the sensitivity of LSTM performance to model compression strategies that reduce the storage and computational costs and compare its performance against (a) alternative deep learning models such as Multi Layer Perceptron (MLP) and (b) shallow machine learning models, such as GMM, SVM and Logistic Regression.

Key Contributions: We make three key contributions:

Performance of Deep Learning Models: We implemented and evaluated an LSTM-based (Long Short Term Memory, a variant of RNN) model for breathing acoustic based authentication (for both user identification and user verification tasks), on three representative devices: smartphone, smartwatch and Raspberry Pi. User identification is a task where we resolve “Which user is attempting to authenticate?” and user verification is a task which involves checking “Is this really user A?”. We also compare performance of LSTM against MLP,

GMM, SVM and Logistic Regression based models. We measured the performance in terms of f-score, model sizes, model loading and inference time and show that LSTM based models for user authentication are smaller in size, lightweight and more accurate than MLP, GMM, SVM and Logistic Regression based models and thus better suited for resource-constrained devices. An uncompressed LSTM model's f-score ranges from 80%–94% and is only 50–180 KB in size (for relevant breathing gestures) and can run on smartphones and smartwatches with approximately 7–10 ms and 18–66 ms of inference timings, respectively. F-score can be increased further by 2%–10% by using majority voting or probability fusion methods, making these methods a compelling approach for improving the security of authentication systems based on only single samples.

Evaluation of Model Compression Methods: We show how model compression techniques such as quantization and factorization can help to further reduce the memory footprint of LSTM and MLP models without impacting performance. Quantized, as well as factorized models, were able to achieve 25%–55% reduction in size compared to the unmodified model for LSTM with the drop in f-score being only 5% in the worst case scenario. The reduction achieved for MLP was in the range of 25%–80% while providing similar f-score as the unmodified models.

Inference performance on mobile CPU and mobile GPU: We evaluated the impact of offloading computational tasks to GPU (Graphical Processing Unit) for LSTM. To take advantage of GPU functionality, we extended MobiRNN [4] framework to obtain models that are suitable for execution on wearables and Raspberry Pi in addition to smartphones. We show that the use of a GPU decreases the inference time (by 50%–300%, depending on the device and the breathing gestures) for LSTM based models. However, as the number of hidden units in the LSTM model increases, the benefits of GPU offloading start to diminish. This is because LSTM models with a higher number of hidden units tend to have more information flowing into the deep neural network which causes memory bandwidth on the device to become a bottleneck.

2 RELATED WORK

As our work is closely related to making deep learned models work on resource-constrained devices, we only focus on the similar theme based works. Lane et al. [19] tried to understand the resource requirements and execution bottlenecks related to CNN and DNN architectures on different devices: mobile, wearable and IOT for audio and vision based applications. Their study showed that complex CNN models have significant resource requirements and hence could not be executed efficiently on these devices. To address this problem, Bhattacharya et al. [3] proposed *SparseSep*. In *SparseSep*, the main idea is to find the sparse representation of fully connected layers and separate the convolutions kernels by creating separate filters. This reduces the number of parameters and the number of convolutional operations required for a deep learning model to operate and hence reduce the computational and space complexity. DeepEar [20] utilized DNN models on smartphones to realize practical audio sensing applications in noisy environments. By implementing DeepEar on DSP of a smartphone, the authors showed that smartphone consumed only 6% of the total energy in a day. DeepEye [25] deployed CNNs on wearables to enable continuous vision applications. It allows local execution of multiple deep vision models by interleaving execution of computation-heavy convolutional layers with the loading of memory-heavy fully connected layers. A similar system named DeepMon [15] enable continuous vision applications on resource-constrained devices by focusing on reducing the processing latency of convolutional layers by doing a set of optimization techniques. Firstly, DeepMon proposed a caching mechanism that leverages similarities between consecutive images. Secondly, DeepMon perform faster matrix multiplication by model decomposition and unfolding. DeepMon also offloads processing of the convolutional layers to GPUs, which accelerates processing.

MobiRNN [4] employs GPU offloading to make LSTM run faster on smartphones for activity recognition tasks using RenderScript.¹ QuantizedCNN [37] introduced quantization of filter kernels in convolutional layers

¹<https://developer.android.com/guide/topics/renderscript/compute.html>

and weighting matrices in fully connected layers to reduce the storage and memory overhead of CNN models on smartphones to allow them to classify images within seconds. In another work, Kim et al. [17] introduced one-shot whole network compression to compress CNN models.

As we see, most of the work to date has focused on improving the performance and deployment of CNNs and DNNs on resource-constrained devices. While CNN is good at delivering excellent results on spatial data, LSTM is more appropriate for sequential data such as audio or text. Hence, there is a clear need to study the performance of LSTM on resource-constrained devices in terms of resource requirements and execution bottlenecks. We took the first step in this direction by investigating the performance of deep learning models, especially LSTM on acoustic data (breathing sounds), which can be used for deploying an end-to-end user authentication solution on mobile, wearable and IoT devices.

3 METHODOLOGY

3.1 Dataset

We used the breathing acoustic samples from *BreathPrint* [7] which presented the feasibility of a breathing acoustics based user authentication system using Gaussian Mixture Models. The original dataset consisted of 70 breathing samples (audio files) per breathing gesture (sniffing, normal breathing and deep breathing) collected from 10 users over three sessions. Each session was 3–4 days apart in time. A sniff gesture consists of two quick consecutive inhalations. A normal breathing gesture has an inhalation followed by an exhalation phase, while a deep breathing gesture involves a long inhalation, followed by a long exhalation.

For our experiments, we divide the dataset into five folds while keeping 80%, 10% and 10% of the total samples for training, validation and testing sets respectively in each fold. If we keep the number of samples for each user in accordance with the time they are collected, the five folds look like as shown in Figure 1. As deep learning models require large volumes of data for training and validation, we created an augmented dataset for both training and validation using the method similar to *BreathRNNNet* [8]. To create an augmented dataset, we used frequency wrapping [16] and amplitude scaling [28]. Each sample was scaled 10 times along the time axis and the amplitude by selecting two separate values from a uniform distribution; $U(0.8, 1.2)$. This leads to an 11-fold increase in the number of samples for both the training and validation set. Note that we do not perform data augmentation on the test set. We focus only on two breathing gestures; sniff and deep, as results from *BreathPrint* [7] showed that these two gestures provide better performance compared to normal breathing gesture.

We created 10 ms frames from each sample using Hamming window smoothing and for each frame, we calculated 96 Mel Frequency Cepstral Coefficients (MFCC) features; 32 MFCC, 32 Delta MFCC and 32 Double Delta MFCC. Then we combine frames to create different windows of different sizes. We used the best window sizes identified in *BreathRNNNet* [8]. Those sizes were 30 and 250 (corresponding to a duration of 30 ms and 2500 ms), respectively for the sniff and deep breathing gesture with 90% overlap. The aforementioned approach is used to create windows for training data, validation data and testing data from breathing samples present in the training, validation and testing set respectively. Note that in the rest of the paper except this subsection the term sample refers to a window as they were used for the final training, validation and testing. To keep the training and validation data balanced, we used the NearMiss-3 undersampling algorithm [24]. We used NearMiss-3 algorithm as it is a controlled undersampling algorithm which allows one to specify the number of windows needed in each class. In each fold, we specify the number of windows for each class to be determined by the number of windows present in class with the least number of windows. Table 1 provides a summary of the dataset for the five folds. Additionally, we also combined consecutive windows from each sample in the test set to create new test sets to perform majority voting and probability fusion based testing (see Section 4.1.1). The size for the number of consecutive windows was chosen to be three and five. For further details on the initial breathing

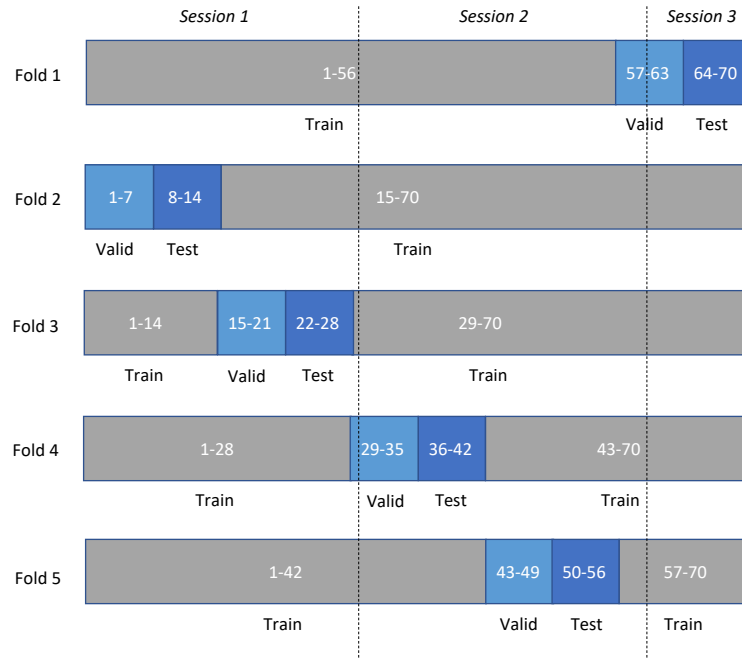


Fig. 1. Folds for Training, Validation and Testing Set. Dotted Vertical Lines Delineates the Different Sessions.

dataset, we encourage the readers to refer to read [7, 8] that describes the data collection, data augmentation and pre-processing methods in more detail.

Note that this work significantly differs from BreathPrint and BreathRNNNet in a number of ways. In BreathPrint, the main idea was to introduce novel behavioural modality, which can be used for user authentication in particular user verification. BreathPrint used GMM classifier on a laptop like a cloud-based solution. In comparison, the focus of this work is to evaluate the performance of different shallow and deep learning models to see if BreathPrint can be efficiently implemented and perform real-time authentication on various resource-constrained devices for both user identification and user verification tasks. The other major difference is the *dataset*. We take the BreathPrint dataset but uses techniques such as random scaling and frequency warping to create a new augmented dataset.

This work also differs from BreathRNNNet, primarily in the scale of evaluation and in our final objective. The major differences lie in the *type of authentication, evaluated classifiers, GPU offloading and extra compression techniques for deep learning models*. While BreathRNNNet only dealt with user identification, this paper investigates both user identification and user verification. Additionally, while BreathRNNNet only evaluated the performance of SVM and LSTM for a fixed configuration (2 layers, 128 hidden units), this paper evaluates the relative performance of other shallow models (MLP, GMM and Logistic Regression) and study the performance of LSTM under more varied architectures and scenarios (e.g., {2,4} layers with multiple hidden units, GPU offloading, and inclusion of factorization and non-linear model compression techniques).

3.2 User Identification

In user identification, we decide whether a breathing sample belongs to a user from a closed set of known users, which can be mapped to a multi-class classification problem.

Table 1. Summary of the Dataset

	Sniff	Deep
Frame size	10 ms	10 ms
Window size	30	250
Overlap	90%	90%
Average Training set size (std. dev.)	22,044 (369)	26,928 (4039)
Average Validation set size (std. dev.)	2,354 (564)	3124 (1353)
Average Test set size (std. dev.)	460.2 (31)	647.2 (80.52)

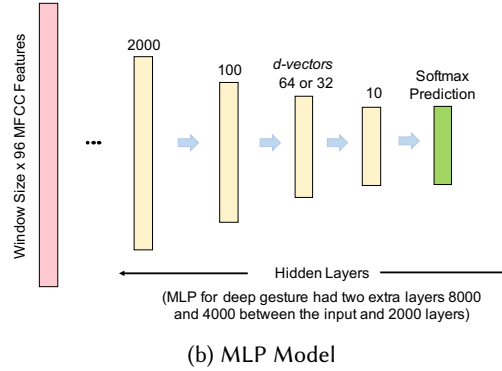
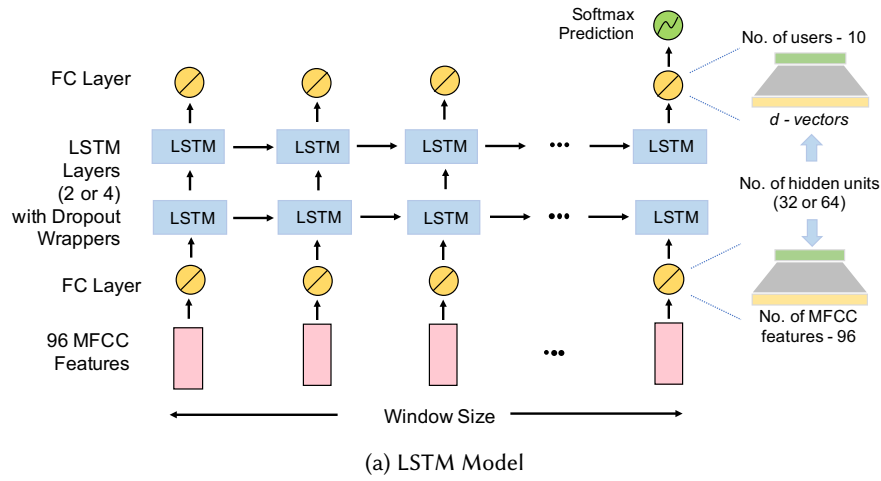


Fig. 2. Deep Learning Network Architectures

3.2.1 Deep Learning Based Approach. We train two deep learning models; LSTM and Multi Layer Perceptron (MLP) with different configurations. For LSTM we used an architecture that is similar to Hammerla et al. [13]. We trained a number of models for combinations of two and four LSTM layers and 32 and 64 hidden unit sizes

(Figure 2a). Each LSTM layer had a dropout wrapper and we tried dropout keep probability (p_{keep}) values between 0.7 and 1.0 at steps of 0.05.

For MLP, we created a network (Figure 2b) that takes as an input size of $window\ size \times 96$ followed by the hidden layer of sizes 2,000, 100, and 32 or 64 and softmax for the sniff gesture. To handle the input sizes in the deep breathing gesture the MLP architecture had two extra layers of size 8000 and 4000 between the input layer and size 2000 layer. We created two networks where the size of the second last hidden layer is either 32 or 64. We tried *tanh*, *ReLU*, and *Sigmoid* activation functions in addition to varying the dropout keep probabilities similar to the LSTM architecture.

All the models were implemented using TensorFlow and trained with up to 1000 epochs using *Adam* optimizer [18] with mini batches of size 32 with early termination. Henceforth, we will use the notation based on *number of layers/number of hidden units* to refer to an LSTM model. For example, an LSTM model with two layers and 32 hidden units would be referred to as LSTM (2/32). Similarly, for MLP we will use the notation based on *number of units in the last hidden layer*. For example, an MLP model with 32 units in the last hidden layer would be referred to as MLP (32). Note that for each classifier, validation set was tested to obtain the tuning parameters. The parameters providing the highest average f-score on all the folds were then used to create the final models for testing the test set. In Table 2 we show the chosen hyper parameters for each classifier.

Table 2. Activation and Dropout Rates of Deep Learning Models

	Sniff		Deep	
Model	Activation	p_{keep}	Activation	p_{keep}
LSTM (2/32)	NA	1.0	NA	0.7
LSTM (4/32)	NA	0.7	NA	0.8
LSTM (2/64)	NA	0.85	NA	0.7
LSTM (4/64)	NA	0.7	NA	0.85
MLP (32)	Sigmoid	0.95	Sigmoid	0.75
MLP (64)	Sigmoid	0.9	Sigmoid	1.0

3.2.2 SVM Based Approach. To compare the performance of deep learning models with conventional machine learning models, we trained a multi-class SVM classifier with a linear kernel and RBF kernel using libSVM [5]. For SVM with linear kernel, the optimal value for penalty C (regularization parameter) was found out from the possible values ranging from 2^{-5} to 2^{15} in the steps of 2. For SVM with RBF kernel, the optimal value for gamma g was found out from the possible values ranging from 2^3 to 2^{-15} in the steps of 2. Finally, the values used for C and g were 2^{-5} and 2^{-15} respectively.

3.3 User Verification

In this scenario users upfront gives their identity and breathing acoustic sample to validate whether the user providing the sample is the legitimate user. This can be mapped to a binary classification problem where an incoming breathing sample is compared against the claiming user's model to see if it is classified as a positive or negative. A positive sample means the user is legitimate and is allowed access and vice versa.

3.3.1 Deep Learning Feature Based Approach. We used a similar approach as Variani et al. [36], where we used the previously trained deep learning model (the one created for user identification) as a feature extractor to train a GMM classifier. The pipeline for the entire user verification process is shown in Figure 3. Firstly the MFCC

based features are extracted from the breathing sample and fed into a deep learning model. Next, d -vectors (of size 1×32 or 1×64) are obtained from the hidden layers of the deep learning models (Figure 2). The d -vectors (deep vectors) are then fed as features into GMM based classifier to determine if the user who is claiming to be user "X" and is trying to gain access to the system is actually user "X". The determination is done by calculating log-likelihood scores and is explained in the next subsection. Figure 4 show a 2-dimensional t-SNE plot [22] of the distribution of the d -vectors obtained from LSTM (2/32) model for sniff breathing gesture. The figure shows that the classes are clearly separable and suited for user authentication.

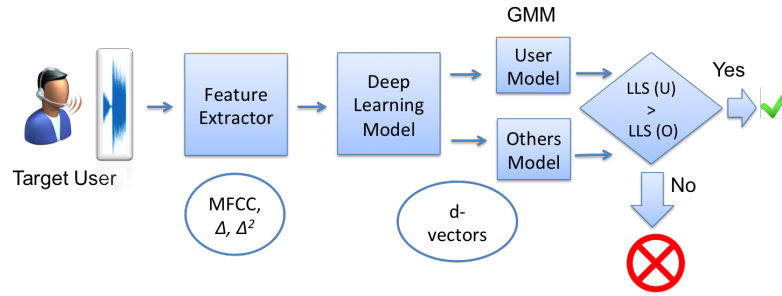


Fig. 3. User Verification–LSTM/MLP, GMM on D-vectors (Deep Vectors)

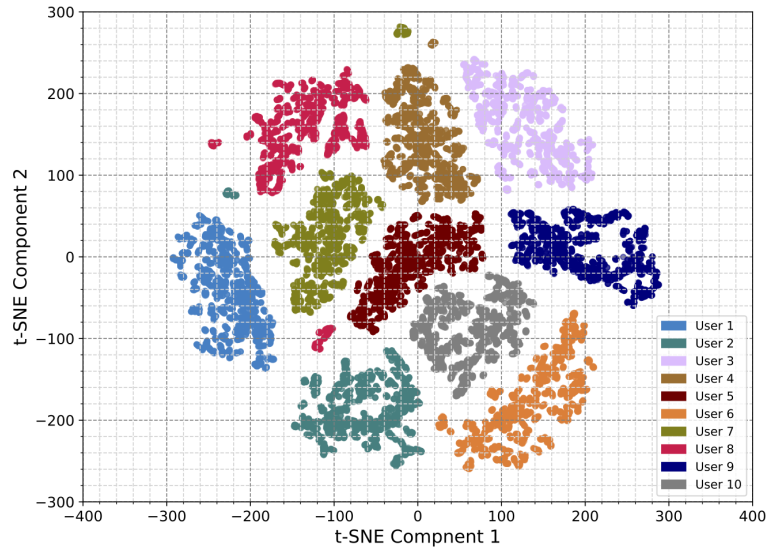


Fig. 4. t-SNE Plot for User verification with D-vectors Obtained from LSTM (2/32) for Sniff Breathing Gesture

3.3.2 MFCC Features Based Approach. To compare the results of deep learning approach with a conventional baseline, we trained a GMM classifier with diagonal co-variances using Scikit. Two models are created for each user. The first model is trained from a target user's data (user's model) while the second model uses data from all other users (other's model). At verification time, an incoming breathing sample is compared against both the models to obtain the log likelihood scores. If the score of the user's model is greater than the other's model then the verification is successful. The GMM based approach uses MFCC features extracted from the breathing samples directly instead of d-vectors. We searched between 1 to 30 to select the correct number of components which provides best results. Two is the optimal number of components for sniff and deep breathing gestures for GMM (only) models. For GMM models based on deep learning, the number of components is five for sniff breathing gestures with LSTM and MLP. The number of components for deep breathing gesture is 30 and 15 for LSTM and MLP respectively with GMM. We also compared the performance of GMM with a model based on Logistic Regression with a penalty l_2 , regularization parameter C of value 1 and *sag* (Stochastic Average Gradient) solver.

3.4 Model Compression

We used two methods; *quantization* and *matrix factorization of fully connected layers* to compress the trained deep learning models. We apply quantization and factorization at different levels as described below to investigate the trade-off between the model size and the f-scores.

3.4.1 Quantization. The fully connected layers in the trained LSTM and MLP models act as a matrix multiplication and the addition of bias terms. The weight values are stored as *32 bit single precision floats* in Tensorflow graphs. To reduce the storage size of the models, we quantized the fully connected layers to 8-bits as described by Han et al. [14] which is a type of non-linear quantization. We tried non-linear quantization because they have been proven to be more accurate than linear quantization [26, 38]. We used a simple k-means clustering of the weight values in the fully connected matrix to find 256 centroids and each weight was approximated to the closest centroid value. This allows us to decrease the size of the matrix by a factor of four. For example, the first hidden layer of our MLP architecture for sniff breathing gesture is a $2,880 * 2,000$ matrix. Thus, it takes approximately 22.5 MB ($2,880 * 2,000 * 4 \text{ bytes}$) of memory. Finding k-means clusters for $k = 256$ allows us to replace *32 bit floats* with *8 bit integers* and gives a four times decrease in the model size.

3.4.2 Matrix Factorization. We used Singular Value Decomposition (SVD), a factorization method that allows decomposing a matrix W into three components U , Σ , and V where Σ is a diagonal matrix. Taking only the first k diagonal elements from Σ allows to get a good representation of the original matrix and this helps to reduce the memory required to store the matrix as well as to reduce the computation time by reducing the number of multiplications. Equation 1 shows the mathematical formulation for SVD.

$$W = U \sum_{k \times k} V^T \quad (1)$$

$n \times d$ $n \times k$ $k \times k$ $k \times d$

The storage gain and the computational gain achieved using SVD are shown in Equation 2 and 3 respectively.

Storage gain specifies how much storage space can be saved when SVD is applied to the original matrix. Higher values of storage gain lead to smaller sized models compared to the unmodified model and thus results in saving space on the device. Similarly, computational gain defines how much speed up can be achieved in terms of the number of matrix multiplications on a compressed model using SVD against doing matrix multiplications on the unmodified model. More precisely, computational gain shows how much faster processing can be done on deep learning models to lower the inference timings. The importance of parameter k (number of SVD components) can be assessed from both the equations. A larger value of k would lead to diminishing gains in both the storage and computation and vice versa. Note the storage gain can be k if the original matrix is sparse.





$$\text{storage gain} = \frac{nd}{nk + k^2 + kd} \quad (2)$$

$$\text{computational gain}_{\text{multiplication}} = \frac{n^2d}{n^2k + k^3 + k^2d} \quad (3)$$

3.5 Performance Metrics

We evaluate the performance of the models on four representative devices listed in Table 3; Pixel 1, Nexus 6P, LG G Watch R and Raspberry Pi 3. Deep learning models are implemented using Tensorflow [1]. MFCC feature extraction is done using Java Speech Toolkit (JSTK).² SVM training is done using LibSVM, while the testing for Android utilize Libsvm-androidjni.³ GMM and Logistic Regression training is done using SciKit [30] and GMM inference code from SciKit was ported to Android. For different parameter configurations, we measured the *f-score*, *model loading time*, *inference time* and *model size*. The average along with standard deviation is reported for all the measured metrics except *model loading time* and *inference time* or when the standard deviation was very small (<0.5%). For loading and inference times we show the measured time from the fold with worst f-score which was the same for all the experiments and the breathing gestures. Typical f-score value ranges between 0 and 1 and is the harmonic mean between the precision and recall. We report f-score results multiplied by 100. Higher f-score signifies better performance of the system. We also measured the *feature extraction time*.

Table 3. Hardware Configuration of the Used Devices

Name	OS	CPU	GPU	Memory
Pixel 	Android 8.0	2.15 GHz Quad-Core	Adreno 530 (650 Mhz)	4 GB
Nexus 6P 	Android 6.0	4x1.55 GHz, 4x2 GHz - Octa-Core	Adreno 430 (630 MHz)	3 GB
LG G Watch R 	Android Wear 2.0	1.2 GHz Quad-Core	Adreno 305 (450 MHz)	512 MB
Raspberry Pi 3 	Android Things 5.0	1.2 GHz Quad-Core	Broadcom VideoCore IV (400 MHz)	1 GB

4 RESULTS

In this section, we discuss the performance of LSTM and MLP deep learning models (uncompressed and compressed) on sniff and deep breathing gestures on multiple devices. Unmodified (*U*) is the original uncompressed model. There are three types of compression strategies applied to the uncompressed model. The first type of compression is *Fact* – *n*, (denoted by F–*n*) where the number *n* indicates the level of factorization showing the number of elements we select in the Σ matrix. The values of *n* were 5, 10, 20, 40, 60, 80 and 100. The next type of compression includes a Quantized only model (*Q*). The final type is *Quant* + *Fact* – *n*, denoted by Q+F–*n*,

²<https://code.google.com/p/jstk/>

³<https://github.com/cnbuff410/Libsvm-androidjni>

which implements quantization with different levels of factorization. As a baseline, we compared the performance of deep learning models with SVM for user identification and with GMM and Logistic Regression for user verification. The feature extraction time for all the devices is in the range of 40–60 ms for the sniff and in the range of 126–484 ms for deep breathing gesture. As expected, Pixel (the most powerful platform) and the smartwatch (least powerful platform), impose the least and highest feature extraction times, respectively.

4.1 User Identification

4.1.1 Baseline: SVM. Contrary to our expectations, the size of SVM model is very large. For linear kernel based SVM the average model size for sniff is 198.6 MB and 2.8 GB for deep breathing gesture. Similarly, the average model size for sniff is 585.4 MB for RBF kernel based SVM. We were not able to create RBF kernel based SVM models for the deep breathing gesture as the training process never converged to a solution. The large size of SVM models is due to a large number of support vectors needed to support the multi-class classification. With linear kernel based SVM model we obtained an f-score of 83.64% and 57.36% for the sniff and deep breathing gesture respectively. The f-score for sniff breathing gesture with RBF kernel based SVM is 84.23%. We find the f-score of the linear and RBF kernel based SVM models to be similar. The model loading time varied between 5,383–30,729 ms while the inference time varied between 244–3,112 ms on all the devices for the sniff breathing gesture on linear kernel based svm models. Although model loading time is very high for SVM based model, it does not impact real-time requirements of the authentication process as model loading is a one time step and can be performed during the initial booting of the device. However, as the memory footprint of SVM models is very high especially on smartwatch with only 512 MB of memory, the performance of other apps or processes running on the device can get impacted. Very large size of models in other cases made it impossible for them to work on all the devices and hence were not tested.

Majority Voting and Probability Fusion Testing: As discussed briefly in Section 3.1, we also created two additional test sets. For creating the first test set, three consecutive windows (beginning from first) from each testing sample was taken. The next consecutive three windows were taken from the window next to the last used window. This process was repeated until no three consecutive windows could be formed. For the second set, the aforementioned process was performed with the number of windows fixed as five. After creating the two test sets, we tested the performance of the classifiers on the test sets in two scenarios. The first scenario is of majority voting and the other one is of probability fusion. In majority voting, consecutive samples are tested in batches of three or five depending on the test set. In the case of three, two of the samples should be correctly identified or verified to make the final decision as positive. While in the case of five, at least three samples should be correctly identified or verified. In probability fusion scenario [9, 23], the probability scores from each class were added for all the samples in a batch (three or five). The class with the maximum average score is predicted as the output. The majority voting and probability fusion scenario were performed to see if the f-score of the classifiers can be improved by not only relying on a single testing sample and thereby obtaining a more secure system. We will refer to majority voting with three and five samples as $MV - 3$ and $MV - 5$ respectively in the rest of the paper. Accordingly, probability fusion will be referred to as $PF - 3$ and $PF - 5$.

F-scores with majority voting on user identification cannot be calculated and hence we only report PF based results. The results for probability fusion methods for SVM show that on an average 5%–10% increase in f-score can be achieved compared to testing based on a single sample. A higher increase is seen for deep breathing gesture. However, at the same time increased values of f-score comes at the expense of higher inference times. The inference time for testing based on three samples and five samples is three times and five times higher than single sample testing. Provided that inference timing with SVM models is already high for single samples, the inference time based on three or five samples would be quite high especially if a user wants to get authenticated quickly.

4.1.2 LSTM. Figure 5, Figure 6 and Table 4 shows the f-score and size respectively for different LSTM models for the test set. We show only results for fewer models to ensure better readability and save space. The f-score achieved with sniff (deep) breathing gesture is around 92%–94% (80%) for the unmodified models. Combining information from the figures and the tables, we can make the following observations:

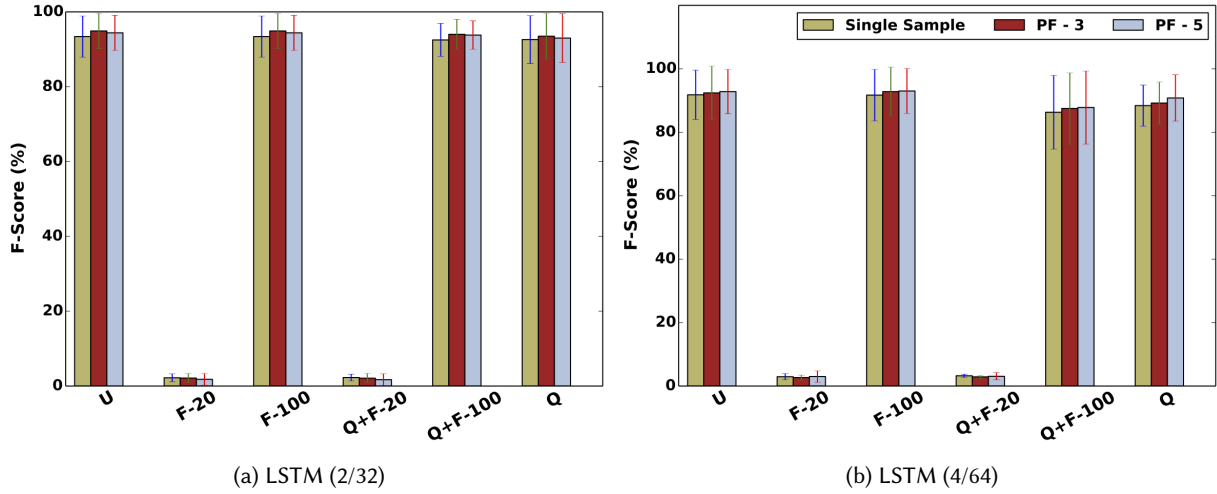


Fig. 5. Average F-score for User Identification - LSTM for Sniff Breathing Gesture

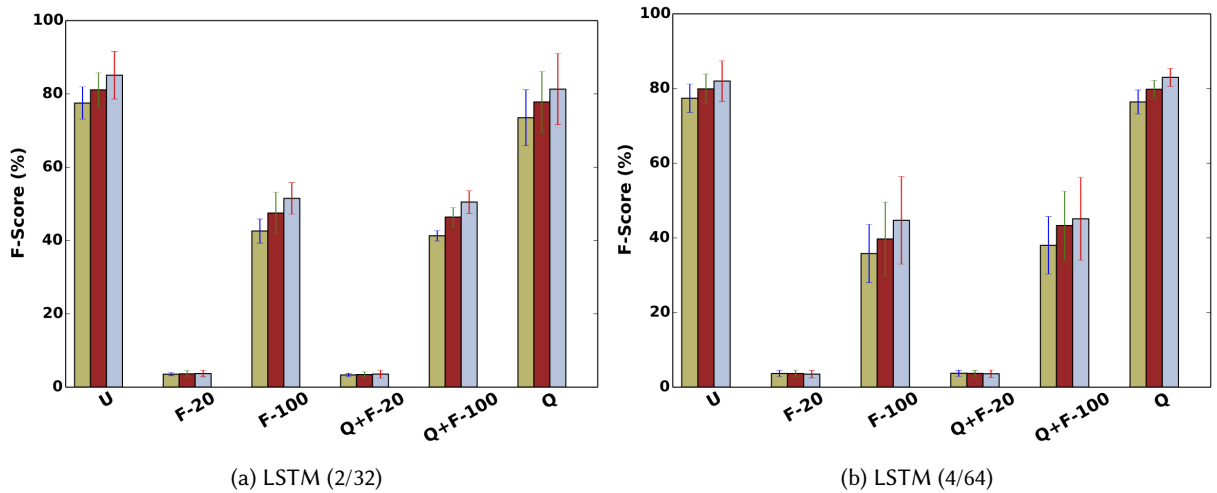


Fig. 6. Average F-score for User Identification - LSTM for Deep Breathing Gesture

- (1) Compared to SVM, the LSTM models are much smaller in size (ranging from 20 KB to 180 KB) and achieve higher f-score for both the breathing gestures. *This suggests that LSTM models do not have high memory requirements, can be loaded even on devices with minuscule memory and achieve good performance,*

Table 4. Average Model Size in MB (Std. Dev.) - User Identification - LSTM for Breathing Gestures. Sizes for Fact - 40, Fact - 60, Fact - 80 is Similar to Fact - 20 Model. Sizes for Quant + Fact - 40, Quant + Fact - 60, Quant + Fact - 80 is Similar to Quant + Fact - 20 Model.

Gesture	Sniff				Deep			
Layers	2		4		2		4	
Hidden Units	32	64	32	64	32	64	32	64
Unmodified	0.05	0.15	0.05	0.15	0.08	0.18	0.08	0.18
Fact - 20	0.04	0.13	0.04	0.13	0.04	0.13	0.04	0.13
Fact - 100	0.05	0.17	0.05	0.16	0.05	0.16	0.05	0.16
Quant + Fact - 20	0.02	0.06	0.02	0.06	0.02	0.06	0.02	0.06
Quant + Fact - 100	0.03	0.07	0.03	0.07	0.03	0.072	0.03	0.076
Quant only 256	0.03	0.07	0.03	0.07	0.06	0.1	0.06	0.1
SVM - Linear	198.6 (6.8)				2800 (489.9)			
SVM - RBF	585.4 (15.4)				-			

- (2) With increasing number of layers and hidden units, the LSTM model size tend to increase (2–3 fold) as the number of deep learning parameters increase with increasing network complexity. However, in general, this seems to have minimal impact on the f-scores,
- (3) Compressed version of LSTM model (Quant + Fact - 100 and Quantized only) can achieve similar f-score as the unmodified model yet with smaller memory footprint. The size of models with compression: Quant + Fact - 100 and Quantized only are 40%–54% smaller in size compared to the unmodified model for sniff breathing gesture with the drop in f-score being 5% in worst case. Similarly, the reduction achieved for deep breathing gesture on compressed models are 25% to 60% depending on the number of hidden units. Higher reduction is seen for models with more hidden units. With increasing complexity, the number of parameters gets larger and hence provide more opportunities to the compression methods to reduce the size of the model.
- (4) Compressed models which are only Quantized or Quantized as well as Factorized are better than models compressed with only Factorization. They are better because they provide similar f-scores as the unmodified models with smaller model sizes. Compressed models with only Factorization have similar memory footprint as the unmodified models. *Observations (2), (3) and (4) suggest that compressed models (Quantized only or Quantized and Factorized) and with lesser complexity (smaller number of layers and hidden units) should be preferred over more complex and unmodified models as they have smaller memory footprint but achieve similar (high) f-scores.*
- (5) The impact of probability fusion methods is different for the sniff and deep breathing gesture. As the f-scores achieved with sniff breathing gesture for single sample testing is already very high, the improvement with PF methods is minimal (1%–2%). However, the same methods help to improve the f-scores by 5%–10% over single sample testing for deep breathing gesture.

The model loading time and inference time for the three devices for two layers with 32 hidden units and four layers with 64 hidden units are shown in Figure 7 and Figure 8 as they represent the best and worst case for LSTM model complexity. The times for Nexus 6P are in similar range as Pixel and hence not shown in the paper. The following observations hold on the basis of results:

- (1) The inference time on Pixel is 7 ms (9 ms), 18 ms (46 ms) on smartwatch and 11 ms (28 ms) on Pi for an unmodified LSTM model with two layers and 32 hidden units for the sniff (deep) breathing gesture. The inference time is increased by 5%–40% as the number of layers are increased from two to four across devices and different breathing gestures. Unlike SVM where the inference time was quite high for a single sample testing, better f-scores can be achieved by the system with PF method at the expense of slight increase in inference time for LSTM models. Note that it would still remain below one second. *These timings are much smaller than SVM based model (244–3,112 ms) and shows that running LSTM model on all the three devices is suitable for real-time user authentication,*
- (2) Compression has negligible impact on the inference time and model loading time,
- (3) Increasing the number of hidden units or the number of layers has minimal impact on model loading time. *Hence, running less complex (small number of layers and small hidden units) and compressed models is most effective on resource-constrained devices.*

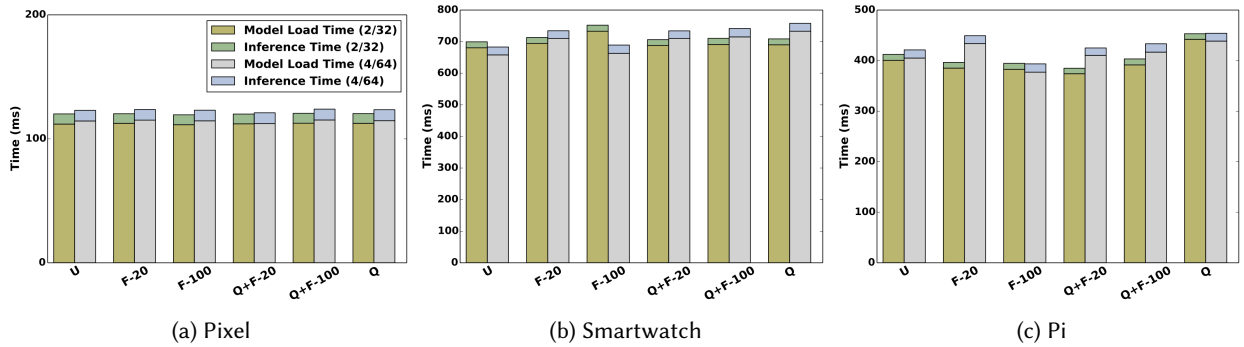


Fig. 7. Model Loading and Inference Times for Three Devices for Sniff Breathing Gesture on LSTM Models

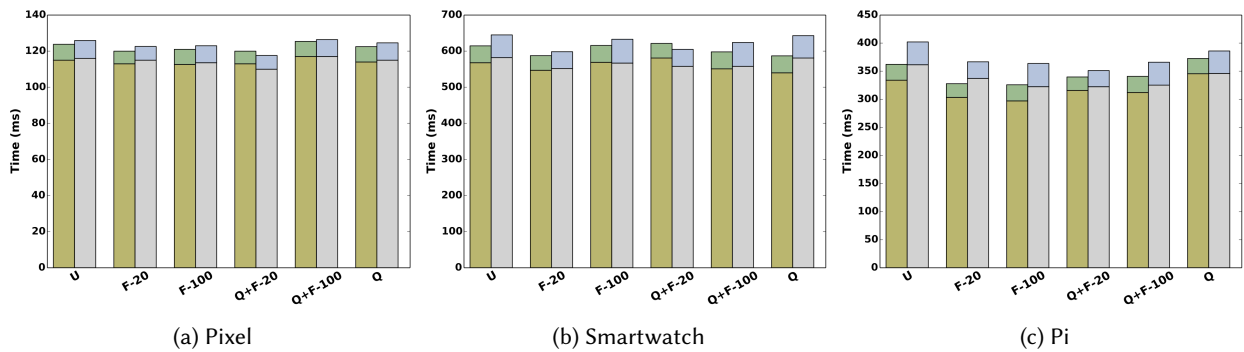


Fig. 8. Model Loading and Inference Times for Three Devices for Deep Breathing Gesture on LSTM Models

4.1.3 MLP. Figure 9 and Table 5 shows the f-scores and model sizes respectively for different MLP models for sniff breathing gesture. While Figure 10 shows model loading and inference timings. Only MLP (32) models for deep breathing gesture could be evaluated and the results are shown in Table 5 and Figure 11. Further, F - 80 and Q + F - 80 were the best (highest sized) models which could be executed on all the devices for MLP (32) for deep breathing gesture. The training on MLP (64) for deep breathing gestures did not converge. Results show that,

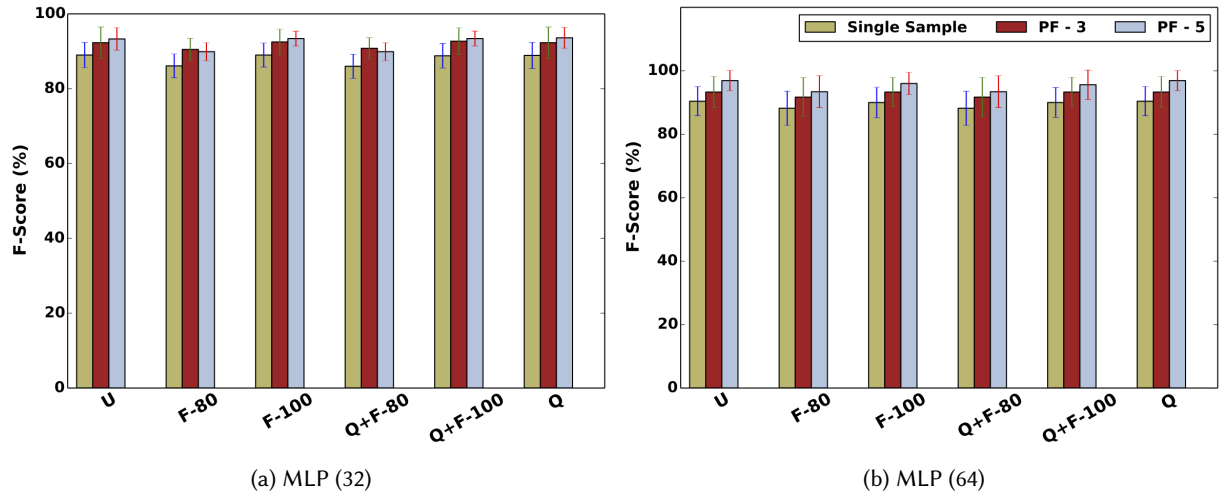


Fig. 9. Average F-score for User Identification - MLP for Sniff Breathing Gesture

- (1) The best f-score achieved with either uncompressed or compressed models for sniff breathing gesture is around 90% which is slightly less than LSTM based model (94%) but higher than SVM based models (83%–84%). The highest f-score achieved by MLP for deep breathing gesture is 63%–64% which is lower than LSTM based models (80%). Also, model sizes for MLP (0.03–21 MB) are smaller than SVM based models (200–600 MB) but bigger than LSTM models (0.02–0.17 MB) for sniff breathing gesture. The model sizes of MLP for deep breathing gesture ranges from 0.05 MB to 184 MB,
- (2) Like LSTM, compression on MLP allows models to have smaller memory footprints and provide similar f-scores as the unmodified models. The size of the compressed models: Quant + Fact - 80 and higher, Fact - 80 and higher and Quantized only are 46%–77% (53%–80%) smaller in size compared to the unmodified model but provide similar f-score as the unmodified model for sniff (deep) breathing gesture. The size of such compressed models ranges from 4.8–21 MB for sniff and 31–80 MB for deep breathing gesture. *These results show that unlike LSTM models which can operate on devices with less than 1 MB of memory, the best MLP models need more than 4 MB of memory which might be unavailable on devices such as MediaTek MT2621,*
- (3) Unlike LSTM based models, compression significantly impacts the inference time of MLP models. On one hand side, a compressed model such as Quant + Fact - 5 can be 5–20 times faster than the unmodified model on all the devices. On the other hand, a compressed model such as Quant + Fact - 100 can be two times slower than the unmodified model on all the devices. Similarly, compression impacts model loading time as well.
- (4) The inference time for MLP models is always higher than LSTM models in general. For example, inference time for an LSTM (2/32) unmodified model is 8 ms, 11 ms and 18 ms on Pixel, Raspberry Pi and smartwatch

Table 5. Model Size (MB)–User Identification with MLP for Different Breathing Gestures

Gesture	Sniff		Deep		Sniff		Deep
Layers	32	64	32		32	64	32
Unmodified	21.09	21.11	170	Quant + Fact - 5	0.02	0.02	0.05
Fact - 5	0.03	0.04	0.11	Quant + Fact - 10	0.03	0.03	0.09
Fact - 10	0.06	0.03	0.19	Quant + Fact - 20	0.05	0.05	0.16
Fact - 20	0.1	0.1	0.36	Quant + Fact - 40	0.1	0.11	0.54
Fact - 40	0.21	0.23	1.23	Quant + Fact - 60	1.67	1.69	8.25
Fact - 60	3.84	3.89	19.16	Quant + Fact - 80	4.94	4.85	31.12
Fact - 80	11.38	11.21	72.28	Quant + Fact - 100	15.29	15.27	79.75
Fact - 100	35.3	35.3	184.8	Quant only 256	9.07	9.04	73.31

respectively for sniff breathing gesture. While, the inference timings for MLP (32) model are 14 ms, 72 ms and 97 ms for the three devices in the same order. The exception to the aforementioned observation happens with highly compressed MLP models. A Fact - 5 MLP (32) model can perform inference faster (only 3.5 ms) than LSTM (2/32) models but with f-score of 11% which is not suitable for any form of authentication. Likewise, a Quant + Fact - 60 MLP (64) model can achieve higher f-score (64% vs. 13%) and lower inference time (6 ms vs. 8 ms) compared to a similar Quant + Fact - 60 LSTM (4/64) model but at the expense of a much bigger model size (1.69 MB vs. 0.06 MB). Note that Quant + Fact - 60 MLP (64) still fails to surpass the performance obtained with unmodified LSTM (2/32) model. Similar results hold for deep breathing gesture. *This result suggests although some very highly compressed MLP models can provide better inference time and f-scores than similarly compressed LSTM models, such models still fail to surpass the f-scores, model size and inference timings achieved for best LSTM models.*

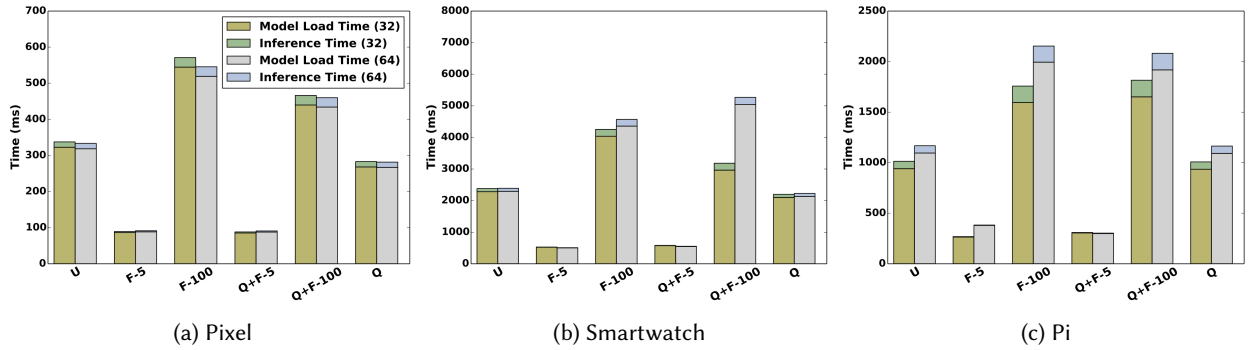


Fig. 10. Model Loading and Inference Times for Three Devices for Sniff Breathing Gesture on MLP Models

Overall, we can conclude that in terms of f-scores and minimal resource requirements, LSTM is superior to SVM and MLP.

4.1.4 Statistical Significance Results. We did statistical significance test on single samples testing f-scores to check if the f-scores between the classifiers (unmodified models) does not happen due to a chance (randomness).

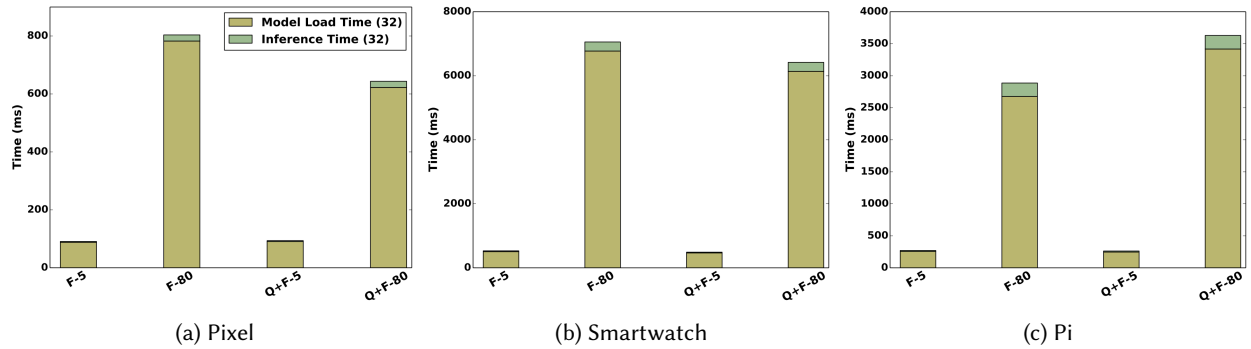


Fig. 11. Model Loading and Inference Times for Three Devices for Deep Breathing Gesture on MLP Models

As we have five different folds (datasets) and more than five different classifiers to test for significance testing, we applied a non-parametric Friedman test [11] followed by a posthoc test named Nemenyi test [10] at 0.05 level of significance. We chose a non-parametric test because it does not depend on the assumptions for data to be following a certain type of distribution. When many folds or datasets are present and many classifiers are to be compared, a Friedman test is performed to check the hypothesis that there exists no significant differences between the performance of classifiers is true or not. If the Friedman test shows that a significant difference exists between the classifiers then a pairwise Nemenyi test is performed to find the classifiers between which significant differences exist. After applying Friedman or Nemenyi test, the obtained value is compared against a critical value to check for the significant difference at different significance level. For the difference to be significant, the value obtained must be higher than the critical value which can be fetched from standard statistical tables.

Sniff: After applying the Friedman test we found the value of Friedman statistic to be 22.75 which is higher than the critical value of 15.5 (significance level 0.05 and the number of compared classifiers is eight). This shows that the performance based on f-scores is indeed significantly different between the classifiers. Further analysis based on Nemenyi test is shown in Table 6. The critical value to pass the Nemenyi test is 3.03. The entries marked in bold in the Table 6 shows that the performance differences between the two classifiers fail to cross the critical value of 3.03 and is hence not significant.

Deep: After applying the Friedman test we found the value of Friedman statistic to be 25 which is higher than the critical value of 14.07 (significance level 0.05 and the number of classifiers is seven). This shows that the performance based on f-scores is indeed significantly different between the classifiers. Further analysis based on Nemenyi test is shown in Table 6. The critical value is 2.94.

The overall results suggest that the difference in f-scores between the SVM's, between the MLP's and between the LSTM's are in general not significantly different. However, the significance between the performance is apparently clear between the SVM, MLP and the LSTM models. LSTM should be favoured upon SVM or MLP and deep learning classifiers perform better than shallow machine learning classifiers for user identification tasks.

4.2 User Verification

4.2.1 Baseline (GMM). The size of the trained GMM model with two GMM components for sniff is 2.16 MB and 17.7 MB for deep breathing gesture.. GMM models for sniff could achieve a f-score of 86.24%. With majority voting and probability fusion based methods the f-score further increases by 3%–5%. The model loading time varied between 1,951–16,222 ms while the inference time varied between 27–134 ms on all the devices. GMM model for deep breathing gesture could achieve an f-score of 74.08%. With majority voting and probability fusion based

Table 6. Statistical Significance –User Identification for Breathing Gestures. Bold Entries Means No Significance.

Classifier	SVM - Linear		SVM - RBF		MLP (32)		MLP (64)		LSTM (2/32)		LSTM (4/32)		LSTM (2/64)		LSTM (4/64)	
Gesture	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep
SVM - Linear			1.29		7.75	4.7	8.39	6.22	16.14	13.91	16.78	16.83	14.52	18.30	10.01	16.83
SVM - RBF					6.45		7.1		14.85		15.49		13.23		8.7	
MLP (32)							0.65	1.46	8.39	9.15	9.04	12.08	6.78	13.54	2.26	12.08
MLP (64)									7.75	7.69	8.39	10.61	6.13	12.08	1.61	10.61
LSTM (2/32)											0.65	2.93	1.61	4.39	6.13	2.93
LSTM (4/32)													2.26	1.46	6.78	0.00
LSTM (2/64)															4.52	1.46
LSTM (4/64)																

methods f-score further increases by 0.5%–2%. The model loading time varied between 12,936–110,769 ms while the inference time varied between 148–1,086 ms on all the devices. The results with GMM show that although sniff has small memory overhead and small inference time, the deep breathing gesture imposes significant overhead (high memory footprint and 1 second of inference time).

4.2.2 (Logistic Regression). The size of the trained LR models for sniff is 240 KB and achieve a f-score of 84.52%. As LR models did not provide better f-scores than GMM models for sniff breathing gesture, we did not evaluate them further.

4.2.3 LSTM–GMM. The verification process for deep learning models works in two steps: (a) Features called d-vectors are extracted from the second-last layer of LSTM or MLP model and then (b) d-vectors are fed to the GMM classifier to obtain final classification output. As a result, the total inference time consists of time to obtain d-vectors from the deep learning model and obtaining outcome from the GMM classifier which adds some extra latency. Similarly, extra storage is required to store GMM model. The number of d-vectors depends on the number of hidden units in LSTM or the number of hidden layers in MLP i.e. 32 or 64. The number of components used in the GMM classifier is five for the sniff and 30 for deep breathing gestures. GMM classifier used diagonal covariance. Details on d-vectors and verification process were presented in Section 3.3.1. The f-scores are shown in Figure 12. Note that we do not test all the models as only a few models showed f-scores comparable to the unmodified models during the user identification tasks. We can make the following observations from the figure and the data analysis done on model sizes and the inference timings:

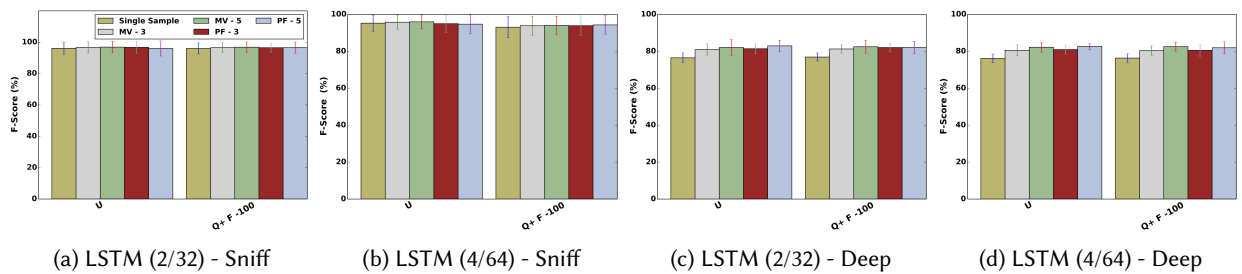


Fig. 12. Average F-score for User Verification - LSTM–GMM for Breathing Gestures

- (1) Verification yields high f-scores for both uncompressed and compressed models. MV or PF based methods helped to increase the f-scores for sniff breathing gesture by 1% or less and by 5%–6% for deep breathing gesture. *This suggests that d-vectors obtained from deep learning models are good at discrimination and can be used as features to train a shallow classifier. Also compressed models providing similar performance as uncompressed models,*
- (2) The inference time is not significantly different between identification and verification for any device for the same number of layers either for an unmodified or compressed model. *This suggests that the overhead added by GMM is minimal in terms of time.* In fact, the additional time introduced by GMM is in range of 1.8 ms (4.4 ms) for Pixel, 3.6 ms (17.7 ms) on smartwatch and 2.5 ms (9 ms) for Pi for sniff (deep) breathing gesture for a 32 d-vector. With 64 d-vectors, the additional time increases roughly 50–100% but is small,
- (3) Like inference time, the extra storage required to perform verification is slightly higher (few KBs) than identification. Note that for verification, the system has to store both deep learning and GMM model. The GMM model size for 32 and 64 d-vectors for sniff breathing gesture is around 80 KB and around 150 KB respectively. The size roughly increases by five times for deep breathing gestures. *This suggests that the overhead added by GMM is minimal in terms of space.,*
- (4) The performance obtained by LSTM–GMM models is better compared to baseline GMM model. For sniff breathing gesture, an unmodified LSTM–GMM (2/32) model achieved an f-score of 96.11 with a model size of 130 KB (50 KB + 80 KB) compared to an f-score of 86.24 obtained for GMM with a model size of 2.16 MB. Also, for the same gesture, the inference time varied between 27–134 ms for GMM baseline model compared to 10–23 ms for LSTM–GMM (2/32) model on all devices. Similar trend is observed for deep breathing gesture.

4.2.4 MLP-GMM. The observations for MLP-GMM models stays same as LSTM–GMM models. Note that MLP–GMM models could only be evaluated with MLP (32) for deep breathing gesture. Verification obtains high f-scores for both unmodified and compressed models (see Figure 13) in case of MLP-GMM. The f-scores achieved with MLP–GMM are similar to LSTM–GMM. As MLP–GMM models for sniff breathing gesture uses GMM classifier (number of components is five) with the same configuration as that of LSTM–GMM models the extra overhead due to GMM is similar in terms of storage requirements and latency which is very small. For deep breathing gesture, the storage requirement and inference timing for MLP-GMM is half of that of LSTM-GMM as the number of GMM components were half. Despite this the total inference time for verification with MLP–GMM models is higher than LSTM–GMM models. This is because the MLP models in general (c.f. Section 4.1.3) always provide higher inference time than LSTM models as shown in previous sections and the overhead due to GMM is very small. Also, LSTM models (KBs) are smaller than MLP based models (MBs). *Less storage requirements, better inference timings along with similar performance results suggest that LSTM–GMM models should be preferred over MLP–GMM models for user verification like user identification.* With respect to the GMM baseline model, MLP-GMM models achieved a mixed bag of results. In general, MLP-GMM models can obtain high f-scores than GMM baseline model at the cost of larger model sizes. The inference timings can be higher or lower for MLP-GMM models compared to GMM baseline model depending on the level and type of compression.

Overall, we can say that like identification, verification also yields good performance (high f-scores) at the expense of very small extra memory footprint (few KBs) and extra processing time (few ms) introduced by GMM.

4.2.5 Statistical Significance Results. We did statistical significance testing on f-scores with five different folds (datasets) and different classifiers in case of user verification. When applying Friedman test we found the value of Friedman statistic to be 27.26 (15.51) which is slightly higher than the critical value of 15.5 (12.59) at a significance level of 0.05 for sniff (deep) breathing gesture. This shows that the performance based on f-scores is indeed significantly different between the classifiers. Further analysis based on Nemenyi test is shown in Table 7. The critical value is 3.03 and 2.85 for the sniff and deep breathing gestures respectively.

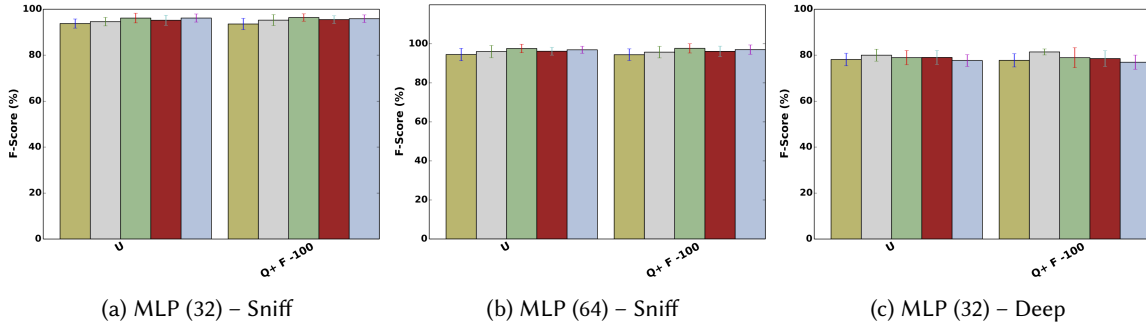


Fig. 13. Average F-score for User Verification - MLP-GMM for Breathing Gestures

Overall, the results for user verification are similar to that of user identification except for one observation. For deep breathing gesture, the MLP models were not found to be significantly different than LSTM based models and provide slightly better f-scores (as seen in previous subsections). However, due to the mentioned reasons at the end of the last subsection, LSTM should be picked over MLP for user verification tasks.

Table 7. Statistical Significance –User Verification for Breathing Gestures. Bold Entries Means No Significance. For Sniff and Deep Breathing gesture the Number of Compared Classifiers is Eight and Six Respectively.

Classifier	GMM		MLP (32)		MLP (64)		LSTM (2/32)		LSTM (4/32)		LSTM (2/64)		LSTM (4/64)		LR	
Gesture	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep	Sniff	Deep
GMM			7.75	18.59	8.39		16.14	8.45	15.49	10.14	17.43	5.07	10.33	5.91	0.65	
MLP (32)					0.65		8.39	10.14	7.75	8.45	9.68	13.52	2.58	12.67	8.39	
MLP (64)							7.75		7.1		9.04		1.94		9.03	
LSTM (2/32)									0.65	1.69	1.29	3.38	5.81	2.53	16.78	
LSTM (4/32)											1.94	5.07	5.16	4.22	16.13	
LSTM (2/64)													7.1	2.53	0.84	
LSTM (4/64)															10.97	
LR																

4.3 GPU Results

We investigate what kind of performance can be achieved by using GPU on LSTM models as these models have shown the best performance among all the compared classifiers. Note that TensorFlow in Android only supports CPU. MobiRNN framework [4] provides GPU support by using RenderScript on smartphones. We used and extended the MobiRNN framework for smartwatch and Raspberry Pi. Figure 14 shows the results of running unmodified LSTM models on different devices. It shows the speed-up ratio: CPU/GPU. It is estimated by dividing inference time obtained using CPU by the inference time obtained using GPU. We make following observations:

- (1) In general, the performance of GPU improves over CPU when there is an increase in the number of layers while keeping the same number of hidden units. The speedup achieved is 2–4 times on all the devices. With an increasing number of layers, the number of deep learning parameters increases but not enough to saturate the memory bandwidth. Also, as the model complexity increases with increasing number of layers the parallelization offered by GPU helps in increasing the speedup.

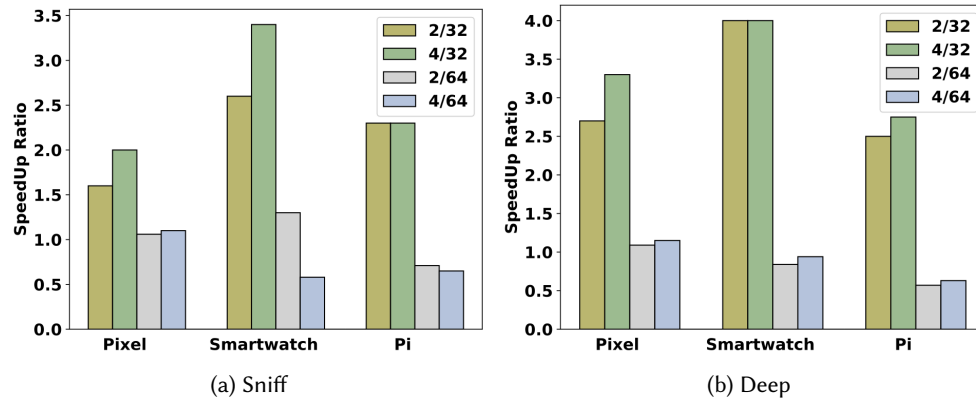


Fig. 14. GPU/CPU SpeedUP Ratio on Multiple Devices for Different Breathing Gestures on LSTM Model

- (2) As the number of hidden units goes higher from 32 to 64, the speed-up diminishes drastically for the smartphones, while for smartwatch and Raspberry Pi the CPU tends to outperform GPUs (GPU on smartwatch and Pi are of lower quality than GPU on smartphones). The reason for diminishing performance is the fact that with more hidden units more information is flowing in the deep learning network and the memory bandwidth of the device becomes the bottleneck, which decreases the amount of parallelization in GPUs.
- (3) Devices with tighter resource constraints such as smartwatch and Pi get more benefit from using GPU than smartphones as long as memory bandwidth does not become the bottleneck. This might be because Pixel already possesses a very powerful CPU and hence the benefits of GPU get offset when compared to smartwatch or Pi.

These results suggest that running less complex models (smaller number of layers and hidden units) on GPU of resource-constrained devices yield greater benefits than running the same models on CPU or running more complex (higher number of layers and hidden units) models on GPU.

5 DISCUSSION

While our work establishes the feasibility of executing LSTM-based classifiers on embedded devices, there are still open issues to the use of an LSTM-based breathing acoustics user authentication.

5.1 Practicality of Breathing Based Authentication

Our results have shown that breathing based authentication is practical on resource constrained devices, providing best-case f-scores of around 96–97% for sniff breathing gestures and 83% for deep breathing gestures. However, this performance may not be high enough for standalone primary authentication, especially compared to fingerprint based authentication which typically have f-scores lower than 1%. Still, breathing-based authentication can be an attractive option for ubiquitous applications:

- (1) Breathing based authentication can be used as a second form of authentication in a multi-factor authentication [29]—for example, when combined with passwords or face recognition [33]. As an example, consider a use case involving the use of a sensitive tool in a factory environment. While passwords or face recognition can provide an initial entry-point authentication (when the worker picks up the tool), breathing can serve

as the secondary mechanism for subsequent, periodic or *continual* user verification. Breathing-based approaches have the advantage that they are effortless and non-intrusive, and also does not require specialized hardware such as fingerprint or iris scanners.

- (2) The level of acceptable authentication performance also depends on the type of application. For example, the level of security needed to perform financial transactions needs to be much higher than managing personalized interactions in a smart home. In scenarios where a very high degree of security is not required (e.g., in an elderly smart home where the attack likelihood is low), breathing based biometrics can work as a standalone weak biometrics [2] to support services such as dispensing daily medication.

5.2 Scalability

The scalability of authentication systems is an important issue, especially when deployed on a non-personal IoT device (e.g., a factory equipment with multiple users). Scalability is less of an issue on a personal device (e.g., smartphone), as the platform is typically concerned only with *user verification*. In particular, an increase in the number of users would potentially negatively affect the training time, accuracy, model sizes and the inference times of an authentication mechanism. We discuss these issues one by one.

- (1) Impact on Training Time: With increasing number of users, the system has to be retrained with the new data coming from the new users. The naive approach of retraining with the entire dataset would require progressively increasing time and computational resources. This can be tackled by the use of transfer learning and incremental learning approaches, widely used in machine learning literature [31, 35]. Incremental learning helps existing system to learn new examples from the same classes, and could be used when the system receives training data from an existing user pool. In contrast, transfer learning helps an existing system to accommodate new classes and could be used to bootstrap the mechanism for new users.
- (2) Impact on Performance: The increasing number of users will impact accuracy and the resource requirements of any authentication system in a negative manner, and the *BreathPrint* system is likely to be no exception. However, there are some possible ways to improve the system's scalability. Multimodal or multifactor authentication systems offers a reasonable way to maintain a high accuracy (security) with increasing number of users. Such systems has some extra user overhead (multiple input actions) and have higher inference times than the standalone systems, but provide higher security guarantees as the number of users grows. An alternative way to increase the scalability is to create multiple models, each consisting of a fixed number and set of users (i.e., partitioning the database or system). The number of users can always be decided based on some acceptable threshold on either the model size or the authentication latency. To combat the issue of decreasing accuracy with an increased number of users, the set of users in a single model can be chosen such that their behavioural biometrics are highly dissimilar. Techniques based on distance similarities or knn can be used to identify such user sets.

6 CONCLUSION

Our study shows that an LSTM based approach for user authentication on breathing acoustics is better than both traditional shallow approaches (SVM, GMM) and a more practical approach than deep learning approach based on MLP because of smaller inference timings, smaller memory footprint and similar performance. LSTM models are robust, provide good f-scores for a secondary authentication modality and are also lightweight (a couple of hundred KB) so that they can run effectively on a variety of resource-constrained devices. The time to authenticate a user is very small (≤ 200 ms) for smartphones and is within acceptable bounds (≤ 1 second) for the highly resource-constrained smartwatch platform. Additionally, compressed models can provide similar f-scores as unmodified (original) model and thus help to reduce storage and memory footprint on such devices. The inference time for LSTM models can be improved 2–4 times by using GPUs instead of CPUs. To conclude,

our investigations suggest that LSTM models offer a compelling lightweight authentication solution based on breathing acoustics to authenticate users in an unobtrusive and natural way on mobile, wearable and IoT devices without connecting to the network.

REFERENCES

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. Usenix, Savannah, GA, USA, 265–283.
- [2] Heikki Ailisto, Elena Vildjiounaite, Mikko Lindholm, Satu-Marja Mäkelä, and Johannes Peltola. 2006. Soft biometrics—combining body weight and fat measurements with fingerprint biometrics. *Pattern Recognition Letters* 27, 5 (2006), 325–334.
- [3] Sourav Bhattacharya and Nicholas D Lane. 2016. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *SenSys*. ACM, Stanford, USA, 176–189.
- [4] Qingqing Cao, Niranjan Balasubramanian, and Aruna Balasubramanian. 2017. MobiRNN: Efficient Recurrent Neural Network Execution on Mobile GPU. In *EMDL (EMDL '17)*. ACM, Niagara Falls, USA, 1–6.
- [5] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 27.
- [6] Jagmohan Chauhan, Hassan Jameel Asghar, Anirban Mahanti, and Mohamed Ali Kaafar. 2016. Gesture-based continuous authentication for wearable devices: The smart glasses use case. In *International Conference on Applied Cryptography and Network Security*. Springer, Guildford, UK, 648–665.
- [7] Jagmohan Chauhan, Yining Hu, Suranga Seneviratne, Archan Misra, Aruna Seneviratne, and Youngki Lee. 2017. BreathPrint: Breathing Acoustics-based User Authentication. In *MobiSys*. ACM, Niagara Falls, USA, 278–291.
- [8] Jagmohan Chauhan, Suranga Seneviratne, Yining Hu, Archan Misra, Aruna Seneviratne, and Youngki Lee. 2018. Breathing-Based Authentication on Resource-Constrained IoT Devices using Recurrent Neural Networks. *Computer* 51, 5 (2018), 60–67.
- [9] Robert T Clemen. 1989. Combining forecasts: A review and annotated bibliography. *International journal of forecasting* 5, 4 (1989), 559–583.
- [10] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* 7, Jan (2006), 1–30.
- [11] Milton Friedman. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association* 32, 200 (1937), 675–701.
- [12] Gartner. 2017. Gartner Says Worldwide Wearable Device Sales to Grow 17 Percent in 2017. <https://www.gartner.com/newsroom/id/3790965>. (2017).
- [13] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables. In *AAAI*. AAAI Press, Phoenix, USA, 1533–1540.
- [14] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).
- [15] Loc N Huynh, Youngki Lee, and Rajesh Krishna Balan. 2017. DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications. In *MobiSys*. ACM, Niagara Falls, USA, 82–95.
- [16] Md Tamzeed Islam, Bashima Islam, and Shahriar Nirjon. 2017. SoundSifter: Mitigating Overhearing of Continuous Listening Devices. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, Niagara Falls, USA, 29–41.
- [17] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. 2015. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530* (2015).
- [18] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [19] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, and Fahim Kawsar. 2015. An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In *International Workshop on IoT-App*. IEEE, Seoul, South Korea, 7–12.
- [20] Nicholas D. Lane, Petko Georgiev, and Lorena Qendro. 2015. DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments Using Deep Learning. In *UbiComp*. ACM, Osaka, Japan, 283–294.
- [21] Aloysius Low. 2017. Mobile payments to overtake credit cards by 2019, says UN report. <https://www.cnet.com/news/come-2019-users-will-prefer-to-pay-by-mobile-instead-of-credit-cards-says-un-report/>. (2017).
- [22] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
- [23] Spyros Makridakis and Robert L Winkler. 1983. Averages of forecasts: Some empirical results. *Management Science* 29, 9 (1983), 987–996.
- [24] Inderjeet Mani and I Zhang. 2003. kNN approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, Vol. 126. Washington, USA, 1–7.

- [25] Akhil Mathur, Nicholas D. Lane, Sourav Bhattacharya, Aidan Boran, Claudio Forlivesi, and Fahim Kawsar. 2017. DeepEye: Resource Efficient Local Execution of Multiple Deep Vision Models Using Wearable Commodity Hardware. In *MobiSys*. ACM, Niagara Falls, USA, 68–81.
- [26] Daisuke Miyashita, Edward H Lee, and Boris Murmann. 2016. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025* (2016).
- [27] Netimperative. 2017. Health and fitness app usage grew 330% in just 3 years. <http://www.netimperative.com/2017/09/health-fitness-app-usage-grew-330-just-3-years/>. (2017).
- [28] Dzung Tri Nguyen, Eli Cohen, Mohammad Pourhomayoun, and Nabil Alshurafa. 2017. SwallowNet: Recurrent neural network detects and characterizes eating patterns. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*. IEEE, Hawaii, USA, 401–406.
- [29] Lawrence O’Gorman. 2003. Comparing passwords, tokens, and biometrics for user authentication. *Proc. IEEE* 91, 12 (2003), 2021–2040.
- [30] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.
- [31] Syed Shakib Sarwar, Aayush Ankit, and Kaushik Roy. 2017. Incremental Learning in Deep Convolutional Neural Networks Using Partial Network Sharing. *arXiv preprint arXiv:1712.02719* (2017).
- [32] Suranga Seneviratne, Yining Hu, Tham Nguyen, Guohao Lan, Sara Khalifa, Kanchana Thilakarathna, Mahbub Hassan, and Aruna Seneviratne. 2017. A survey of wearable devices and challenges. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2573–2620.
- [33] Terence Sim, Sheng Zhang, Rajkumar Janakiraman, and Sandeep Kumar. 2007. Continuous verification using multimodal biometrics. *IEEE transactions on pattern analysis and machine intelligence* 29, 4 (2007), 687–700.
- [34] Statista. 2018. Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions). <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>. (2018).
- [35] Prahalathan Sundaramoorthy, Gautham Krishna Gudur, Manav Rajiv Moorthy, R Nidhi Bhandari, and Vineeth Vijayaraghavan. 2018. HARNet: Towards On-Device Incremental Learning using Deep Ensembles on Constrained Devices. In *EMDL*. ACM, Munich, Germany, 1–6.
- [36] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. 2014. Deep neural networks for small footprint text-dependent speaker verification. In *ICASSP*. IEEE, Florence, Italy, 4052–4056.
- [37] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized convolutional neural networks for mobile devices. In *CVPR*. IEEE, Las Vegas, USA, 4820–4828.
- [38] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. 2017. Incremental network quantization: Towards lossless cnns with low-precision weights. In *ICLR*. Vancouver, Canada, 1–2.

Received February 2018; revised August 2018; accepted October 2018