Mobile Sensor Data Anonymization

Mohammad Malekzadeh
Queen Mary University of London, UK
m.malekzadeh@qmul.ac.uk

Richard G. Clegg
Queen Mary University of London, UK
r.clegg@qmul.ac.uk

Andrea Cavallaro
Queen Mary University of London, UK
a.cavallaro@qmul.ac.uk

Hamed Haddadi
Imperial College London, UK
h.haddadi@imperial.ac.uk

ABSTRACT

Motion sensors such as accelerometers and gyroscopes measure the instant acceleration and rotation of a device, in three dimensions. Raw data streams from motion sensors embedded in portable and wearable devices may reveal private information about users without their awareness. For example, motion data might disclose the weight or gender of a user, or enable their re-identification. To address this problem, we propose an on-device transformation of sensor data to be shared for specific applications, such as monitoring selected daily activities, without revealing information that enables user identification. We formulate the anonymization problem using an information-theoretic approach and propose a new multi-objective loss function for training deep autoencoders. This loss function helps minimizing user-identity information as well as data distortion to preserve the application-specific utility. The training process regulates the encoder to disregard user-identifiable patterns and tunes the decoder to shape the output independently of users in the training set. The trained autoencoder can be deployed on a mobile or wearable device to anonymize sensor data even for users who are not included in the training dataset. Data from 24 users transformed by the proposed anonymizing autoencoder lead to a promising trade-off between utility and privacy, with an accuracy for activity recognition above 92% and an accuracy for user identification below 7%.

CCS CONCEPTS

• **Security and privacy**; • **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computing methodologies** → **Machine learning approaches**;

KEYWORDS

Sensor Data Privacy, Adversarial Training, Deep Learning, Edge Computing, Time Series Analysis.

ACM Reference Format:
Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. 2019. Mobile Sensor Data Anonymization . In *International Conference on Internet-of-Things Design and Implementation (IoTDI '19), April 15–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3302505.3310068

IoTDI '19, April 15–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6283-2/19/04…$15.00
https://doi.org/10.1145/3302505.3310068

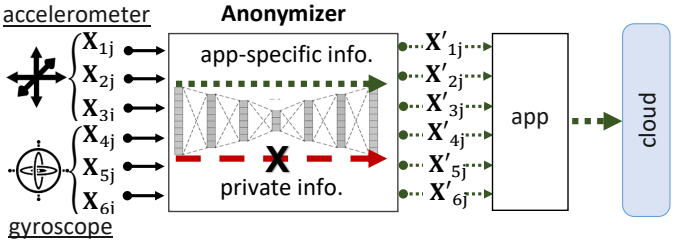

Figure 1: The Anonymizer is a pre-trained autoencoder that transforms raw data before they are shared with an (untrusted) app to enable a service-specific inference that does not reveal private information about the user. KEY - X_{sj}: raw data generated by sensor s at time j; X'_{sj}: corresponding anonymized data after transformation.

1 INTRODUCTION

Motion data from the sensors in mobile and wearable devices can reveal private information about users without their awareness. For instance, motion patterns can be used to create fine-grained behavioral profiles of users that reveal their identity [23]. We are interested in designing an on-device privacy-preserving approach to share with apps transformed sensor data in order to prevent the exposure of sensitive information unrelated to the service while simultaneously preserving the service-specific utility (see Figure 1).

Approaches for privacy-preserving data release include differentially private mechanisms [6] and information theoretic frameworks [28]. *Differential privacy* [36] offers a privacy guarantee for access to private datasets, but it is not applicable to continuously released sensor data. In fact, a private mechanism for publishing sensitive data needs to aggregate all users' data [33] and, in our scenario, we do not trust data aggregators. Moreover, we want to run the mechanism on user devices, but the local version of differential privacy [5, 16] is unsuitable in this case. Time series such as sensor data present recurring patterns in consecutive temporal windows and, unless considerable noise is added to each window that would eliminate the utility of the data, applying the same differentially private mechanism to all windows does not provide a privacy guarantee [32]. Instead, frameworks based on *information theory* [17] consider as the measure of privacy the mutual information between the released data and the latent information that can be inferred from data. Under this framework we do not necessarily need to design a noise addition mechanism and we can remove or at least reduce private information while keeping useful service-specific information [25].

To design a data release mechanism that simultaneously satisfies utility and privacy constraints, we use adversarial approaches to train deep autoencoders [19]. Using adversarial training [7, 34], we

approximate the mutual information by estimating the posterior distribution of private variables, given the released data. Moreover, we anonymize data locally and define a mechanism that can be shared across users, whereas existing solutions need a trusted party to access user personal data to offer a reliable distortion mechanism [16, 24, 37] or need users to participate in a privacy-preserving training mechanism [1].

We formulate the sensor data anonymization problem as an optimization process based on information theory and propose a new way of training deep autoencoders. Inspired by recent advances in adversarial training to discover from raw data useful representations for a specific task [19], we propose a new multi-objective loss function to train deep autoencoders [22]. The loss function regulates the transformed data to keep as little information as possible about user identity, subject to a minimal distortion to preserve utility, which in our case is that of an activity recognition service.

Unlike other approaches [10, 11, 20, 26, 28, 34], our training process not only regulates the encoder to consider exclusively task-specific features in the data, but also shapes the final output independently of the specific users in the training set. This process leads to a generalized model that can be applied to new data of unseen users, without user-specific re-training. We evaluate the efficiency and utility-privacy trade-off of the proposed mechanism and compare it with other methods on an activity recognition dataset[1].

2 RELATED WORK

Adversarial learning enables us to approximate, using generative adversarial networks (GANs) [9], the underlying distribution of data or to model, using variational autoencoders (VAE) [14], data with well-known distributions. These techniques can be applied to quantify mutual information for optimization problems [7, 11, 24, 34] and can be used to remove sensitive information from latent low-dimensional representations of the data, e.g. removing text from images [7]. An optimal privacy mechanism can be formulated as a game between two players, a privatizer and an adversary, with an iterative minimax algorithm [11]. Moreover, the service provider can share a feature extractor based on an initial training set that is then re-trained by the user on their data and then sent back to the service provider [24, 30].

In our work, we do not assume the existence of a trusted data aggregator to perform anonymization for end users. We assume we only have access to a public dataset for training a general anonymization model. The trained anonymizer should generalize to new unseen users, because it is impractical for all users to provide their data for the training.

The feature maps of a convolutional autoencoder have the ability to extract patterns and dependencies among data points and have shown good performance in time series analysis [38]. Autoencoders compress the input into a low-dimensional latent representation and then reconstruct the input from this representation. Autoencoders are usually trained by minimizing the differences (e.g. mean squared error or cross entropy) between the input and its reconstruction [22]. The bottleneck of the autoencoder forces the training process to capture the most descriptive patterns in the data (i.e. the main factors of variation of the data) in order to generalize the model and prevent undesirable memorization [2, 8]. An effective way to train an autoencoder is to randomly corrupt [35] or replace [21] the original input and force the model to refine it in the reconstruction. In this way, a well-trained autoencoder captures prominent and desired patterns in the data and ignores noise or undesired patterns [35]. Moreover, a latent representation can be learned that removes some meaningful patterns from the data to reduce the risk of inferring sensitive information [21].

Only considering the latent representation produced by the encoder and leaving intact the decoder with information extracted from the training data offer only limited protection [7, 16]. Considering the decoder's output leads to a more reliable data protection [20, 26]. In this paper, we consider outputs from both the encoder and decoder of an autoencoder for data transformation. We also consider a distance function as an adjustable constraint on the transformed data to control the amount of data distortion and help tune the privacy-utility trade-off for different applications.

3 SENSOR DATA ANONYMIZATION

We aim to produce a data transformation mechanism to anonymize mobile sensor data so that the user specific motion patterns, that are highly informative about user's identity, cannot be captured by an untrusted app that has access to the sensor to recognize a set of B required activities. Thus, we consider users' identity, that can be inferred from user specific motion patterns, as their sensitive data. We use the concept of mutual information to quantify how much can be inferred about a particular variable from a data set. We wish to minimize the amount the data changes but remove the ability to infer private information from the data.[2]

3.1 Anonymization function

Let sensor component s (e.g. the z axis value of the gyroscope sensor) at sampling instant j, generate $\mathbf{X}_{sj} \in \mathbb{R}$. Let the time series generated by M sensor components in a time-window of length W, be represented by matrix $\mathbf{X} \in \mathbb{R}^{M \times W}$, with $\mathbf{X} = (\mathbf{X}_{sj})$. Let N be the number of users and $\mathbf{U} \in \{0,1\}^N$ be a variable representing the identity of the user; a one-hot vector of length N, a vector with 1 in the k-th place and 0 in all other places if user k generated the data being considered. Let the current activity that generates $\mathbf{X}$ be $\mathbf{T} \in \{0,1\}^B$; a one-hot vector of length B with the one in position b if the current activity is the b-th activity. Finally, we define the data with the user's identifiable information obscured as the *anonymized sensor data*, $\mathbf{X}'$.

Let $\mathrm{I}(\cdot;\cdot)$ be the mutual information function, $\mathrm{d}(\cdot,\cdot)$ a distance function between two time series[3], $\mathrm{A}(.)$ a data transformation function and $\mathbf{X}$ the data we want to anonymize. We define the fitness function $\mathrm{F}(.)$ as

$$\mathrm{F}\left(\mathrm{A}\left(\mathbf{X}\right)\right) = \beta_i \mathrm{I}\left(\mathbf{U}; \mathrm{A}\left(\mathbf{X}\right)\right) - \beta_a \mathrm{I}\left(\mathbf{T}; \mathrm{A}\left(\mathbf{X}\right)\right) + \beta_d \mathrm{d}\left(\mathbf{X}, \mathrm{A}\left(\mathbf{X}\right)\right), \quad (1)$$

[1] Code and data are available at: https://github.com/mmalekzadeh/motion-sense

[2] As notation we use capital bold-face, e.g. $\mathbf{X}$, for random variables (univariate or multivariate) and lowercase bold-face, e.g. $\mathbf{x}$, for an instantiation; roman typestyle, e.g. I, for operations or functions; lowercase math font, e.g. i, for indexing; and capital math font, e.g. M, for specific numbers such as the size of a vector.

[3] In the specific implementation of this paper, we choose as $\mathrm{d}(\cdot,\cdot)$ the mean squared error, MSE, between raw data and the corresponding transformed data. One can choose any other distance functions based on the tasks at hand.

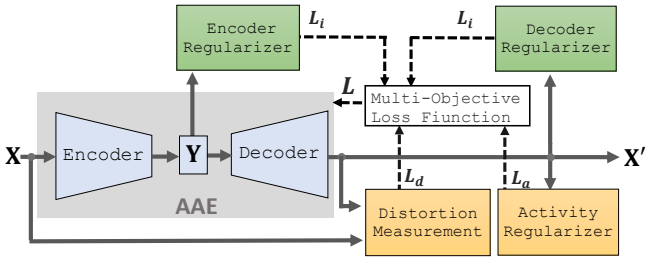

Figure 2: The losses involved in the training procedure. After training, the Anonymizing AutoEncoder (AAE), or Anonymizer, runs on the device as interface between sensor data and (untrusted) apps. KEY – Solid lines: data flow; dashed lines: loss functions. X: raw input data; Y: low-dimensional representation of the input data; X′: transformed data; L_i: identity loss; L_a: activity loss; L_d: distortion loss function; L: overall loss function for training the AAE.

where the non-negative weight parameters β_i, β_a and β_d determine the trade-off between privacy and utility.

Let us define the anonymization function, $\mathcal{A}(\cdot)$, as

$$\mathcal{A}(\mathbf{X}) = \underset{\mathrm{A}(\mathbf{X})}{\operatorname{argmin}} \mathrm{F}\left(\mathrm{A}(\mathbf{X})\right). \tag{2}$$

such that the optimal $\mathcal{A}(\cdot)$ transforms $\mathbf{X}$ into $\mathbf{X}' = \mathcal{A}(\mathbf{X})$, which contain as little information as possible associated to the identity of the user (minimum $\mathrm{I}(\mathbf{U};\mathbf{X}')$), while maintaining sufficient information to discriminate the activity (maximum $\mathrm{I}(\mathbf{T};\mathbf{X}')$) and minimizing the distortion of the original data (minimum $\mathrm{d}(\mathbf{X},\mathbf{X}')$).

As we cannot practically search over all possible anonymization functions, we consider a deep neural network and look for the optimal parameter set through training. To approximate the required mutual information terms, we reformulate the optimization problem in (1) as a neural network optimization problem and train an anonymizing autoencoder (AAE) based on adversarial training.

3.2 Architecture

Let $\mathrm{A}(\mathbf{X};\theta)$ be an autoencoder neural network, where θ is the parameter set and $\mathbf{X}$ is the input vector to be transformed into the output vector $\mathbf{X}'$ with the same dimensions. The network optimizer finds the optimal parameter set θ^* by searching the space of all the possible parameter sets, Θ, as:

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}} \beta_i \mathrm{I}\left(\mathbf{U};\mathrm{A}\left(\mathbf{X};\theta\right)\right) - \beta_a \mathrm{I}\left(\mathbf{T};\mathrm{A}\left(\mathbf{X};\theta\right)\right) + \beta_d \mathrm{d}\left(\mathbf{X},\mathrm{A}\left(\mathbf{X};\theta\right)\right) \tag{3}$$

where, $\mathcal{A}(\cdot;\theta^*)$ is the optimal estimator for a general $\mathcal{A}(\cdot)$ in (1).

We obtain θ^* using backpropagation with stochastic gradient descent and a multi-objective loss function. We also determine values of β_i, β_a and β_d as the trade-off between utility and privacy through cross validation over the training dataset.

Figure 2 shows the framework for the training of the AAE. The Encoder maps $\mathbf{X}$ into an identity concealing low-dimensional latent representation $\mathbf{Y}$ by getting feedback from a pre-trained classifier, the Encoder Regularizer, which penalizes the Encoder if it captures information corresponding to $\mathbf{U}$ into $\mathbf{Y}$. The Decoder outputs

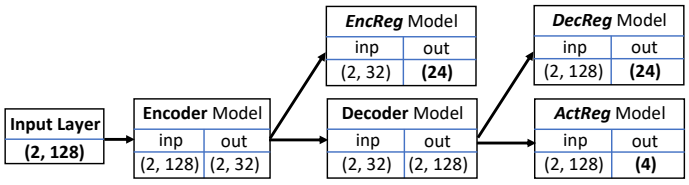

Figure 3: Implementation of the models shown in Figure 2 for a dataset with 24 users and 4 activities. KEY – *EncReg*: Encoder Regularizer; *DecReg*: Decoder Regularizer; *ActReg*: Activity Regularizer.

a reconstruction of the input, $\mathbf{X}'$, from the $\mathbf{Y}$, and gets feedback from other pre-trained classifiers, the Decoder Regularizer and the Activity Regularizer, respectively.

The Encoder Regularizer (*EncReg*) and the Activity Regularizer (*ActReg*) share the same architecture as the Decoder Regularizer (*DecReg*). The only differences are that the shape of input for *EncReg* is 32, instead of 128, and the shape of softmax output for *ActReg* is 4, instead of 24 for a dataset with 24 users and 4 activities. Figure 3 shows the overall architecture whereas Figures 4 and 5 show the details of each neural network model.

Because convolutional layers capture well locally autocorrelated and translation-invariant patterns in time series [15], we choose the two privacy regularizers, *EncReg* and *DecReg*, and the activity regularizer, *ActReg*, to be convolutional neural network classifiers trained by a categorical cross-entropy loss function [38].

3.3 Training

Instead of just training on a single epoch, as usually done in adversarial training [9], all the classifiers should be trained for several epochs, e, on the entire dataset to converge to suboptimal information estimators for use in the next step. In fact, our objective is not to learn the data distribution, but to transform data from an identity-centred sample space (which is informative about users' identity) to an activity-centred sample space (which carry only information about the underlying activity). Therefore, each regularizer should at least converge to a suboptimal approximator of mutual information.

The *EncReg* learns to identify a user among N in the training dataset by getting as input $\mathbf{Y}$ the low-dimensional representation of $\mathbf{X}$ produced by the *Encoder*. The output is the identity label, $\mathbf{U}$. The *DecReg* learns to identify users by getting as input the reconstructed data, $\mathbf{X}'$, produced by the Decoder (here too the output is the identity label, $\mathbf{U}$). The *ActReg* learns to recognize the current activity and gets the reconstructed data, $\mathbf{X}'$, as input and the activity label, $\mathbf{T}$, as output. Finally, the distortion regularizer, a loss function that constrains the allowed distortion on the data, gets the original data, $\mathbf{X}$, and reconstructed data, $\mathbf{X}'$, to calculate pointwise the *mean squared error* to quantify the amount of distortion.

After each iteration, we evaluate the convergence condition of the AAE to decide, based on the current utility-privacy trade-off. We discuss more about possible evaluation methods in Section 4.4.

Figure 6 summarizes the training of the AAE, which can be done locally, on the user powerful devices; centrally, by a service

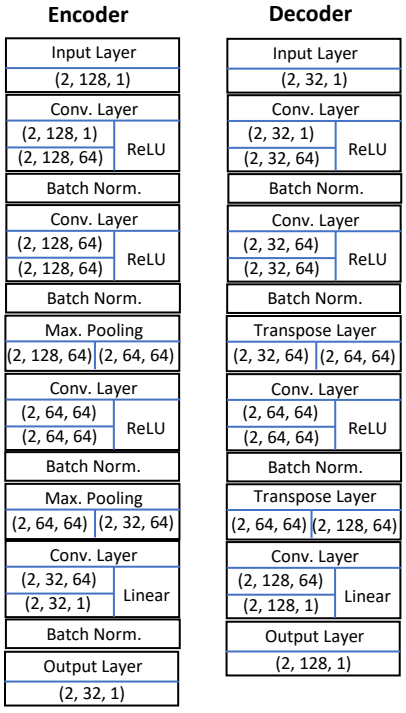

Figure 4: Implementation of the AAE architecture: Encoder and Decoder models in Figure 3.

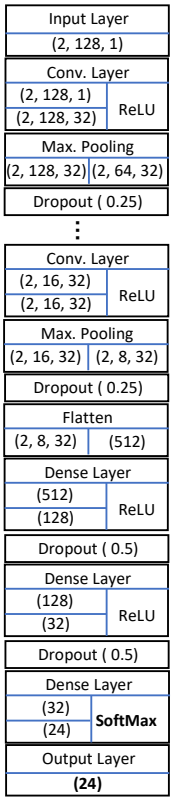

Figure 5: Implementation of the the *DecReg* architecture in Figure 3 (we have the same structure for *EncReg* and *ActReg*).

provider; or a user can download a public pre-trained model and refined it on their own data [29].

3.4 Multi-objective loss function

After each round of training of the regularizers, we freeze their parameters while training the AAE (line 11 of the training procedure, Figure 6). A key contributior to the AAE training is our proposed multi-objective loss function, L, which implements the fitness function $\mathrm{F}\left(\mathrm{A}\left(\mathbf{x}\right)\right)$ of Eq. (1):

$$\mathrm{L} = \beta_i \mathrm{L}_i - \beta_a \mathrm{L}_a + \beta_d \mathrm{L}_d, \tag{4}$$

where the regularization parameters β_a, β_d, and β_i are non-negative, real-valued weights that determine the utility-privacy trade-off. L_a and L_d are *utility losses* that can be customized based on the app requirements (note that L_d is the only available utility loss if there is no target application), whereas L_i is an *identity loss* that helps the AAE remove user-specific signals.

The categorical cross-entropy loss function for classification, L_a, aims to preserve activity-specific patterns[4]:

$$\mathrm{L}_a = \mathbf{T} \log(\hat{\mathbf{T}}), \tag{5}$$

where $\mathbf{T}$ is the one-hot B-dimensional vector of the true activity label for $\mathbf{X}$ and $\hat{\mathbf{T}}$, the output of a softmax function, is a B-dimensional vector of probabilities for the prediction of the activity label.

[4]We can customize L_a for the task e.g. using a binary cross-entropy for fall detection [18].

To tune the desired privacy-utility trade-off, the distance function that controls the amount of distortion, L_d, forces $\mathbf{X'}_{sj}$ to be as similar as possible to the input $\mathbf{X}_{sj}$:

$$\mathrm{L}_d = \frac{1}{M \times W} \sum_{s=1}^{M} \sum_{j=1}^{W} (\mathbf{X}_{sj} - \mathbf{X'}_{sj})^2, \tag{6}$$

Finally, the identity loss, L_i, the most important term of our multi-objective loss function that aims to minimize sensitive information in the data, is defined as:

$$\mathrm{L}_i = -\left(\mathbf{U} \log\left(\mathbf{1}^N - \hat{\mathbf{U}}\right) + \log\left(1 - max\left(\hat{\mathbf{U}}\right)\right)\right), \tag{7}$$

where $\mathbf{1}^N$ be the all-one column vector of length N, $\mathbf{U}$ is the true identity label for $\mathbf{X}$, and $\hat{\mathbf{U}}$ is the output of the softmax function, the N-dimensional vector of probabilities learned by the classifier (i.e. the probability of each user label, given the input).

A trivial anonymization would consistently transform data of a user into the data of another user (and vice versa). However, this transformation would only satisfy the first element of L_i. As no attacker should be able to confidently predict $\mathbf{U}$ from $\mathbf{X'}$, we maximize the difference between the prediction, $\hat{\mathbf{U}}$, and the true identity, $\mathbf{U}$ by minimizing the cross-entropy between the true identity label

procedure TrainAAE($\mathcal{X}, \mathcal{U}, \mathcal{T}, e$) ▷ $\mathcal{X}$: dataset ($M \times W$ temporal windows); $\mathcal{U}$: identity labels; $\mathcal{T}$: activity labels; e number of epochs.
  AAE (*Encoder+Decoder*) ← Random initialization;
  AAE ← Train on $\mathcal{X}$ as both input and output for e epochs;
  $\mathcal{Y}$ ← *Encoder*($\mathcal{X}$); ▷ $\mathcal{Y}$ is the extracted latent representation from the raw data.
  $\mathcal{X}'$ ← *CopyOf*($\mathcal{X}$); ▷ Keep raw data intact to use it for evaluation in each iteration.
  EncReg, *DecReg*, *ActReg*, *AAE* ← Random initialization;
  do
    EncReg ← Train on $\mathcal{Y}$ as input and $\mathcal{U}$ as output using categorical cross-entropy as loss function, for e epochs;
    DecReg ← Train on $\mathcal{X}'$ as input and $\mathcal{U}$ as output using categorical cross-entropy as loss function, for e epochs;
    ActReg ← Train on $\mathcal{X}'$ as input and $\mathcal{T}$ as output using categorical cross-entropy as loss function, for e epochs;
    Freeze parameters of *EncReg*, *DecReg*, and *ActReg*;
    AAE ← Train on $\mathcal{X}'$ as input and $\mathcal{U}$, $\mathcal{U}$, $\mathcal{T}$, and $\mathcal{X}$ as outputs for e epochs (see Figure 2);
    $\mathcal{Y}$ ← Encoder($\mathcal{X}$);
    $\mathcal{X}'$ ← Decoder($\mathcal{Y}$);
    Unfreeze parameters of *EncReg*, *DecReg*, and *ActReg*;
  while it does not satisfies the convergence conditions;
  return AAE; ▷ Resulting AAE to be used as Anonymizer.

Figure 6: The adversarial regularization procedure to train the Anonymizer, $\mathcal{A}(\cdot, \theta^*)$, using Eq. (3)

and the regularizer's prediction of this label, as well as the maximum value of the predicted identity vector, $\hat{\mathbf{U}}$ (see Eq. (7)). The derivation of L_i is presented in the next section.

3.5 Derivation of the identity loss

Our goal is to make $\mathbf{U}$ and $\mathbf{X}'$ independent of each other. To this end, we minimize the amount of information leakage from $\mathbf{U}$ to $\mathbf{X}'$ [25]. As a function f that aims to infer the identity of a user does not increase the available information, the following inequality holds:

$$\mathrm{I}(\mathbf{U}; \mathbf{X}') \geq \mathrm{I}(\mathbf{U}; \mathrm{f}(\mathbf{X}')), \tag{8}$$

and therefore if we reduce the mutual information between the user's identity and their released data, the processing of these data cannot increase the mutual information. The mutual information, $\mathrm{I}(\mathbf{U}; \mathbf{X}')$, can be defined as

$$\mathrm{I}(\mathbf{U}; \mathbf{X}') = \mathrm{H}(\mathbf{U}) - \mathrm{H}(\mathbf{U}|\mathbf{X}'), \tag{9}$$

where $\mathrm{H}(\cdot)$ is the entropy. As the entropy is non-negative and we cannot control $\mathrm{H}(\mathbf{U})$, we maximize the conditional entropy between identity variable and the transformed data, $\mathrm{H}(\mathbf{U}|\mathbf{X}')$, in order to minimize the mutual information, $\mathrm{I}(\mathbf{U}; \mathbf{X}')$:

$$\mathrm{H}(\mathbf{U}|\mathbf{X}') = \mathrm{H}(\mathbf{U}, \mathbf{X}') - \mathrm{H}(\mathbf{X}'). \tag{10}$$

The entropy of $\mathbf{X}'$, $\mathrm{H}(\mathbf{X}')$, can be reduced independently of any other latent variables by simply downsampling the data. However, as blindly minimizing $\mathrm{H}(\mathbf{X}')$ could lead to a substantial utility loss, we focus on maximizing $\mathrm{H}(\mathbf{U}, \mathbf{X}')$.

Let $p(\mathbf{U}, \mathbf{X}')$ be the joint distribution of $\mathbf{U}$ and $\mathbf{X}'$; and $S_{\mathbf{u}}$ and $S_{\mathbf{X}'}$ be the supports of $\mathbf{U}$ and $\mathbf{X}'$, respectively. Then

$$\mathrm{H}(\mathbf{U}, \mathbf{X}') = -\int_{S_{\mathbf{u}}} \int_{S_{\mathbf{X}'}} p(\mathbf{U}, \mathbf{X}') \log p(\mathbf{U}, \mathbf{X}'). \tag{11}$$

We now need an estimator for $\mathrm{H}(\mathbf{U}, \mathbf{X}')$ as we cannot calculate the joint entropy directly for high-dimensional data. When labeled data are available, $\mathbf{X}'$ can be used as input to predict $\hat{\mathbf{U}}$ as an estimation of $\mathbf{U}$. We therefore reformulate the problem of maximizing the joint entropy, $\mathrm{H}(\mathbf{U}, \mathbf{X}')$, as maximization of the cross entropy between the true label, $\mathbf{U}$, and the predicted label, $\hat{\mathbf{U}}$:

$$\mathrm{H}_{\hat{\mathbf{U}}}(\mathbf{U}) = -\int_{S_{\mathbf{X}'}} \mathbf{U} \log \hat{\mathbf{U}}. \tag{12}$$

If $\hat{\mathbf{U}}[k]$ is the k-th element of the vector predicted by the multiclass classifier, the empirical cross entropy for data $\mathbf{X}'$ of user k is:

$$-\mathbf{U} \log \hat{\mathbf{U}} = -\log \hat{\mathbf{U}}[k] \tag{13}$$

and, since $\hat{\mathbf{U}}[k] \in [0, 1]$, maximizing $-\log \hat{\mathbf{U}}[k]$ is equivalent to minimizing $-\log(1 - \hat{\mathbf{U}}[k])$. Therefore minimizing the first term of Eq. (7), $\mathbf{U} \log(\mathbf{1}^N - \hat{\mathbf{U}})$, minimizes the mutual information, $\mathrm{I}(\mathbf{U}; \mathbf{X}')$, and, by forcing the AAE to minimize this value, we minimize the amount of user-identifiable information in $\mathbf{X}'$.

3.6 Examples

To gain an appreciation of the type of distortions introduced by the AAE, we compare sensor data before and after transformation.

Figure 7 (top) shows the low-dimensional latent representation of raw gyroscope data extracted by the bottleneck of the model. The distribution of Y has useful information to distinguish not only the activities, but also the users (color clusters of the top-right plot). Figure 7 (bottom) shows the latent representation of the data anonymized by our method: the transformation masks the data for different users but preserves the Jogging activity samples separated from those of the other activities (note that this is a considerably compressed representation of the input data).

Figure 8 compares raw and transformed data of four activities. It is possible to notice that the AAE obscures patterns and peaks, but maintains differences among data of different activities.

5

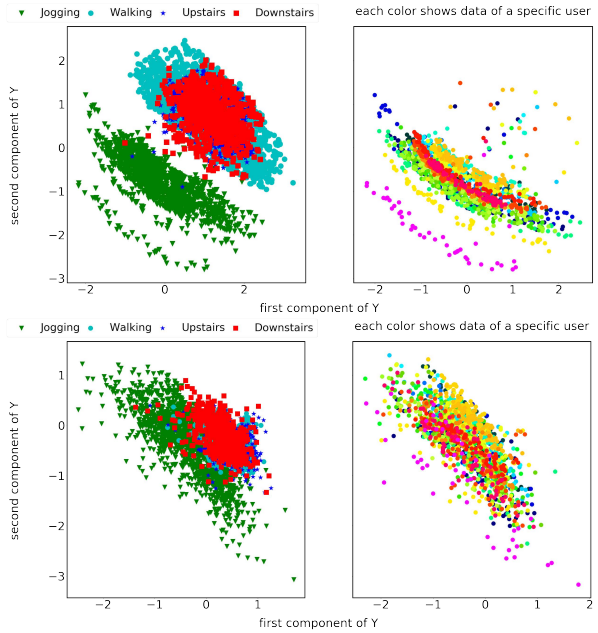

Figure 7: Latent representation, Y, of the 64D gyroscope data in 2D. (Top row): raw data. (Bottom row): data transformed by the AAE. (Left column): samples of four activities. (Right column): Jogging data for all users.

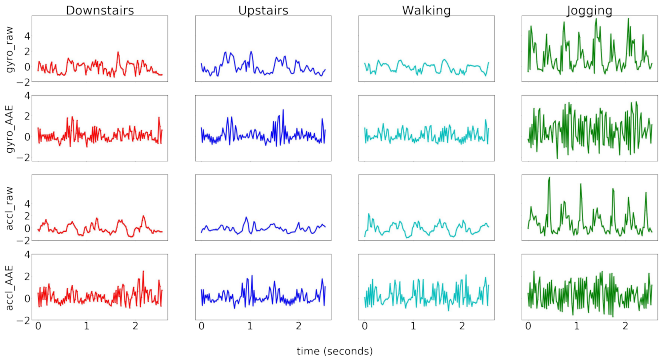

Figure 8: Comparison of raw (first and third row) and transformed data (second and fourth row) for gyroscope (first two rows) and accelerometer (last two rows) for four activities.

Finally, Figure 9 compares the spectrogram of raw and transformed data for a user: the AAE introduces new periodic components and obscure some of the original ones, and they differ across the activities. As periodic components in accelerometer data can disclose information about attributes of users such as height and weight, the AAE reduces the possibility of user re-identification by introducing new periodic components in the data.

In the next section we quantify the performance of the proposed method and compare it with alternative approaches.

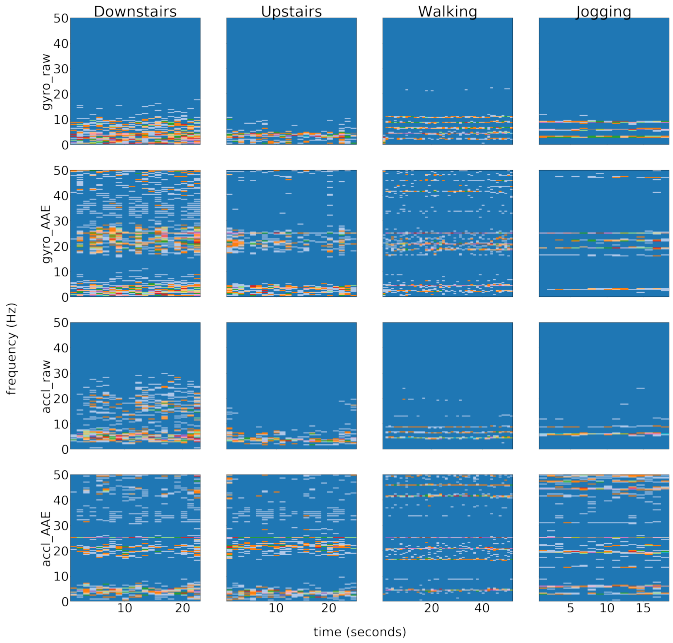

Figure 9: Spectrogram of raw (first and third row) and transformed data (second and fourth row) for gyroscope (first two rows) and accelerometer (last two rows) for four activities.

4 EVALUATION

To evaluate the effectiveness of the proposed data anonymizer, we analyze the trade-off between recognizing the activity of a user and concealing their identity. We measure the extent to which the activity recognition accuracy is reduced by the anonymization process, compared to using the raw data. We compare with two baseline methods for coarse-grained time series data, namely *Resampling* and *Singular Spectrum Analysis* (SSA), and with *REP* [7], which only considers sensitive information included in Y and does not take X′ into account (Figure 2).

4.1 Experimental Setup

Current public datasets of motion sensor data do not simultaneously satisfy the requirements of abundance and variety of activities and users[5]. We therefore collected a dataset from the accelerometer and gyroscope of an iPhone 6s placed in the user's front pocket of tight trousers [13, 20]. The dataset includes 24 participants, in a range of age, weight, height and gender, who performed 6 activities in 15 trials. In each trial, we used the same environment and conditions for all the users (see Table 1). We divide the dataset into training and test sets with two different strategies, namely Subject and Trial. In *Subject*, we use as test data all the data of 4 users, 2 females and 2 males, and as training data that of the remaining 20 users. After training, the model is evaluated on data of 20 unseen users. In *Trial*, we use as test data one trial session for each user and as training data the remaining trial sessions (for example, one trial of Walking of each user is used as test and the other two trials are used as

[5]Datasets that satisfy both (e.g. [23]) are still private.

Number of users	24 (14 males, 10 females)
Sampling rate	50 Hz
Sensors	gyroscope accelerometer
Features	rotationRate (x,y,z) userAcceleration (x,y,z) gravity (x,y,z) attitude(roll, pitch, yaw)
Activities (number of trials)	Downstairs (3 trials) Upstairs (3 trials) Walking (3 trials) Jogging (2 trials) Sat (2 trials) Stand-Up (2 trials)

Table 1: The MotionSense dataset [20]. Multiple trials of the same activity are performed in different locations. KEY – (x, y, z): the three axes of the sensor.

training). In both cases, we put 20% of training data for validation during the training phase. We repeat each experiment 5 times and report the mean and the standard deviation. For all the experiments we use the magnitude value for both gyroscope and accelerometer.

We choose as window length $W = 128$ (2.56 seconds) and we set as stride $S = 10$. For all the regularizers, *EncReg*, *DecReg*, and *ActReg*, we use 2D convolutional neural networks. To prevent overfitting to the training data, we put a Dropout [31] layer after each convolution layer. We also use an L2 regularization to penalize large weights so that the classifier is forced to learn features that are more relevant for the prediction.

4.2 Sensor Data Characteristics

In this section we discuss the characteristics of motion sensor data that informed the design of our sensor data anonymizer.

Figure 10 shows the correlation between the magnitude of the time series collected from these sensors. We see that both sensors almost follow each other, especially for the peaks and periodicity of the magnitude value, whereas a correlation among axes is less obvious. Figure 11 compares the magnitude values of the data from two sensors when the user performs in six different activities. Note that Sat and Stand-Up are difficult to be told apart. The only data that are informative to distinguish these activities from each other are the values of the gravity axes which determine whether the phone is held vertically or horizontally. However, we do not consider Sat and Stand-Up in our experiments for training the AAE.

Figure 12 compares the F1 score obtained using as classifier a deep convolutional neural network with seven groups of data[6]. We

[6]By the similar architecture described in Figure 5

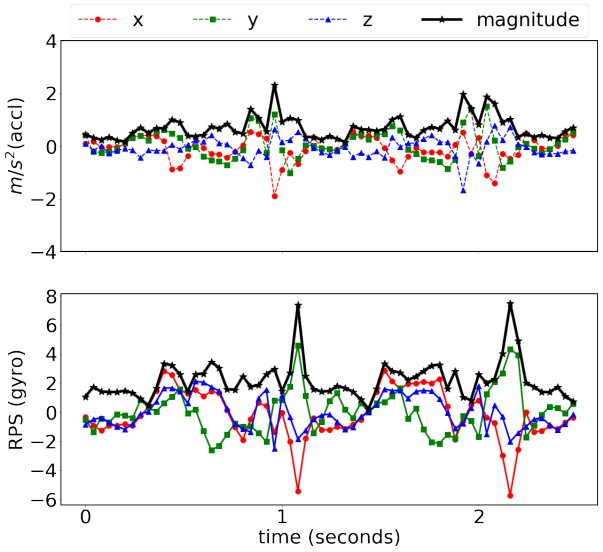

Figure 10: Sample accelerometer (top) and gyroscope (bottom) data for Walking of a specific user. KEY – RPS: revolutions per second, m/s^2: metres per second squared

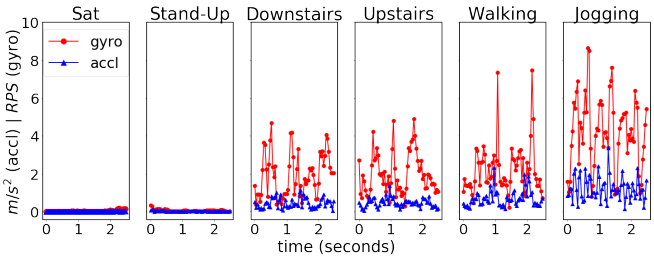

Figure 11: Sample accelerometer (accl) and gyroscope (gyro) data for six activities for a single user.

use the *Subject* setting for activity recognition and the *Trial* setting for identity recognition. The groups of data are the magnitude value of each sensor, the exact value of each axis, the data of only one of these sensors and then both. It is interesting to estimate the amount of information about user's identity that can be extracted form the correlation between accelerometer and gyroscope. Note that we can achieve equal (or better) accuracy for activity recognition using only the magnitude, whereas we should use the values of each axis for identity recognition. Moreover, using a 2D convolutional filter (i.e. the classifier considers the correlation among the input sensors) improves over both activity and identity recognition compared to using 1D filters, which process each input separately. Hence, a good anonymization mechanism should consider both inter-sensor and intra-sensor correlations. We will use the magnitude value of both gyro and accelerometer (*MagBoth_2D*) in our experiments of evaluating the utility-privacy trade-offs.

Figure 13 shows the autocorrelation at varying time lags for the magnitude of accelerometer data for different activities (average over 45 seconds of data for all users). Note that each activity has

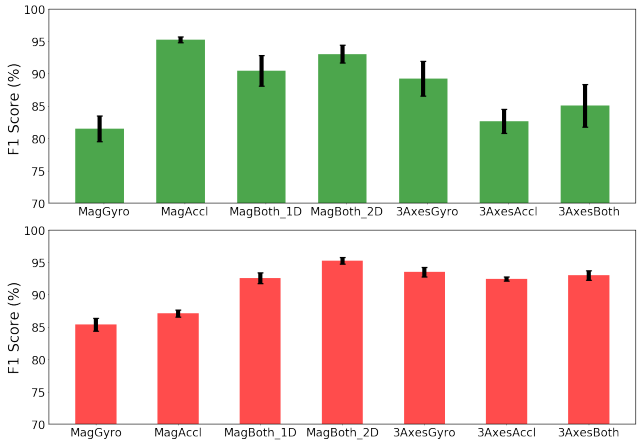

Figure 12: Average F1 score for the recognition, with different sensor data types, of activity (top) and identity (bottom). The black vertical segments show the standard deviation. KEY – Mag: magnitude; gyro: gyroscope; accl: accelerometer; Both: both gyro and accl; 1D and 2D are the dimensions of the convolution filter.

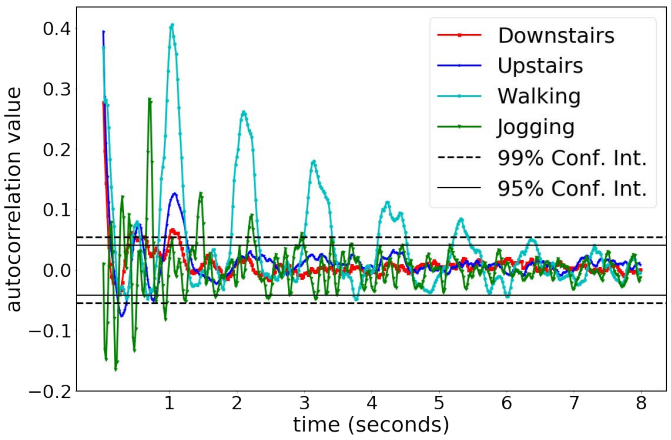

Figure 13: Autocorrelation of accelerometer data for four activities averaged over all the users. Correlation values outside the lines of the confidence interval (Conf. Int.) are statistically significant.

a different period. Walking has the highest correlation, followed by Jogging, Upstairs, and Downstairs. The distance between two peaks can be related to the stride. There are also strong correlations among samples inside a 2-second window, whereas correlations go under the confidence interval after about 5 seconds.

Figure 14 shows the autocorrelations of the same activity performed by three users. The heavier the user, the longer the intervals between two peaks (user **u**_1 is the heaviest among the three). This user-identifiable pattern is a challenging feature to obscure before sharing the data. In fact, we see that baseline methods like downsampling cannot hide the user identity.

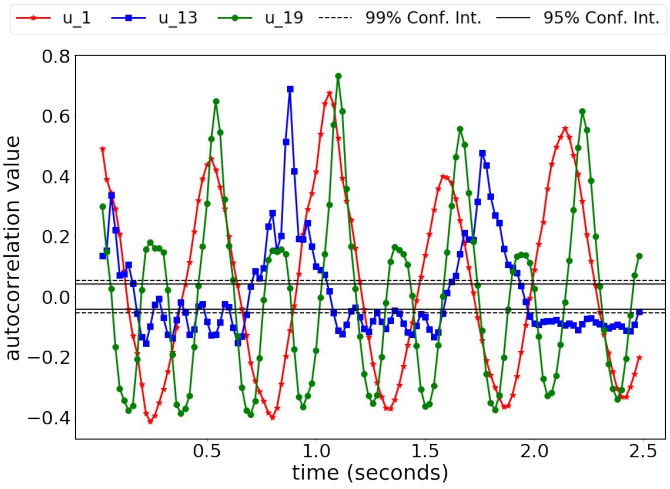

Figure 14: Autocorrelation of the accelerometer data for Walking for three users. KEY – Conf.Int.: confidence interval; u_1, u_{13}, u_{19}: data of user 1, 13, and 19.

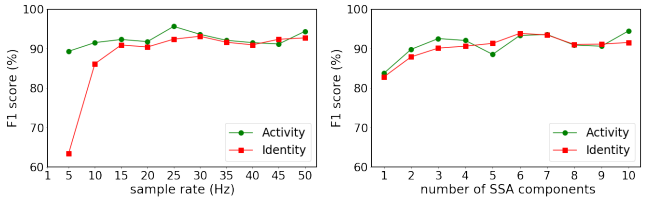

Figure 15: Classification accuracy for a deep convolutional neural network for both Activity and Identity recognition. (Left) Using data resampled to another rate (from 5 to 50 Hz, where 50 Hz is the original sampling rate). (Right) Using data reconstructed using only a subset of components (from 1 to 10, from a total of 50), ordered from largest to smallest by corresponding singular values.

4.3 Baseline Methods

As baseline methods we use Resampling and Singular Spectrum Analysis.

Resampling ideally aims to reduce the richness of the data to the extent that it contains useful information for recognizing the activity but not identity-specific patterns. We choose a resampling based on the *Fast Fourier Transform* (FFT) and, specifically, we use the "signal.resample" function of "SciPy" package [12]. Figure 15 (left plot) shows the classification accuracy with downsampled sensor data. For a fair comparison, we trained a fixed model (in terms of the size of the parameters and number of the layers) for all the sample rates. The impact of downsampling on activity recognition can be ignored for a rates greater than 20Hz. However, even at 5Hz, we can distinguish the 24 users from each other with over 60% accuracy.

Singular Spectrum Analysis (SSA) [3] decomposes time series into interpretable components such as trend, period, and structureless

info.	result	raw (50Hz)	resample (10Hz)	resample (5Hz)	SSA (1,2)	SSA (1)	REP [7] (50Hz)	**AAE** (50Hz)
ACT	mean F1	92.51	91.11	88.02	88.59	87.41	91.47	**92.91**
	var F1	2.06	0.63	1.85	0.91	0.89	00.87	**0.37**
ID	mean ACC	96.20	31.08	13.53	34.13	16.07	15.92	**6.98**
	mean F1	95.90	25.57	8.86	28.59	12.58	11.25	**1.76**
DTW	mean Rank	0	7.2	9.3	6.8	9.5	10.7	**6.6**
	var Rank	0	5.7	5.8	5.6	5.4	5.5	**4.7**

Table 2: Trade-off between utility (activity recognition) and privacy (identity recognition). KEY – ACT: activity recognition, ID: identity recognition, ACC: accuracy, F1: F1 score, DTW: Dynamic Time Warping as the similarity measure, SSA: Singular Spectrum Analysis, REP: Only Anonymizing the latent Representation, AAE: Our Anonymizing AutoEncoder. The forth row shows the K-NN rank between 24 users.

(or noise) components. The window length parameter specifies the number of components. We decompose each $\mathbf{X}$, into a set of D components, $\{\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_D\}$, such that the original time series can be recovered as:

$$\mathbf{X} = \sum_{d=1}^{D} \mathsf{X}_d. \tag{14}$$

As SSA arranges the elements X_d in descending order according to their corresponding singular value, we explore the idea of *incremental reconstruction*. Figure 15 (right plot) shows that training a classifier on the reconstruction with only the first components, up to the total of 10 extracted components, can achieve over 80% accuracy for both activity and identity recognition.

4.4 Discussion

In this section, we compare the transformed data produced by our trained AAE with the outputs of the other methods.

We train an activity recognition classifier on both the raw data and the transformed data, and then use it for inference on the corresponding test data. Here we use the Subject setting, thus the test data includes data of new unseen users. The second row of Table 2 shows that the average accuracy for activity recognition for both Raw and AAE data is around 92%. Compared to other methods that decrease the utility of the data, we can preserve the utility and even slightly improve it, on average, as the AAE shapes data such that an activity recognition classifier can learn better from the transformed data than from the raw data.

To evaluate the degree of anonymity, we assume that an adversary has access to the training dataset and we measure the ability of a pre-trained deep classifier on users raw data in inferring the identity of the users when it receives the transformed data. We train a classifier in the Trial setting over raw data and then feed it different types of transformed data. The third row of Table 2 shows that downsampling data from 50Hz to 5Hz reveals more information than using the AAE output in the original frequency. These results show that the AAE can effectively obscure user-identifiable information so that even a model that have had access to users' original data cannot distinguish them after applying the transformation.

Finally, to evaluate the efficiency of the anonymization with another unsupervised mechanism, we implement the k-Nearest Neighbors (k-NN) with Dynamic Time Warping (DTW) [27]. Using DTW, we measure the similarity between the transformed data of a target user k and the raw data of each user l, $\mathbf{X}^l$, for all $l \in \{1, 2, \ldots, k, \ldots, N\}$. Then we use this similarity measure to find the nearest neighbors of user l and check the rank of k among them. The last row of Table 2 shows that it is very difficult to find similarities between the transformed and raw data of the users as the performance of the AAE is very similar to the baseline methods and the constraint in Eq. (3) maintain the data as similar as possible to the original data.

5 CONCLUSION

We proposed a multi-objective loss function to train an anonymizing autoencoder (AAE) as sensor data anonymizer for personal and wearable devices. To remove user-identifiable features included in the data we consider not only the feature extractor of the neural network model (encoder), but we also force the reconstructor (decoder) to shape the final output independently of each user in the training set, so the final trained model is a generalized model that can be used by a new unseen user. We ensure that the transformed data is minimally perturbed so an app can still produce accurate results, for example for activity recognition. The proposed solution is important to ensure anonymization for participatory sensing [4], when individuals contribute data recorded by their personal devices for health and well-being data analysis.

As future work, we aim to measure the cost of running such local transformations on user devices; to conduct experiments on other use cases (i.e. different tasks); and to derive statistical bounds for the level of privacy protection achieved.

ACKNOWLEDGEMENTS

This work was supported by the Life Sciences Initiative at Queen Mary University London and a Microsoft Azure for Research Award (CRM:0740917). Hamed Haddadi was partially supported by the EPSRC Databox grant (EP/N028260/1). Andrea Cavallaro wishes to thank the Alan Turing Institute (EP/N510129/1), which is funded

by the EPSRC, for its support through the project PRIMULA. We would also like to thank Deniz Gunduz and Emiliano De Cristofaro for their constructive feedback and insights.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 308–318.

[2] Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* 2, 1 (2009), 1–127.

[3] David S Broomhead and Gregory P King. 1986. Extracting qualitative dynamics from experimental data. *Physica D: Nonlinear Phenomena* 20, 2-3 (1986), 217–236.

[4] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. 2006. Participatory sensing. In *Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications*. 117–134.

[5] John Duchi, Martin J Wainwright, and Michael I Jordan. 2013. Local privacy and minimax bounds: Sharp rates for probability estimation. In *Advances in Neural Information Processing Systems*. 1529–1537.

[6] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM, 715–724.

[7] Harrison Edwards and Amos Storkey. 2016. Censoring Representations with an Adversary. In *International Conference in Learning Representations (ICLR2016)*.

[8] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel. 2013. Extracting deep bottleneck features using stacked auto-encoders. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 3377–3381.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[10] Jihun Hamm. 2017. Minimax filter: learning to preserve privacy from inference attacks. *The Journal of Machine Learning Research* 18, 1 (2017), 4704–4734.

[11] Chong Huang, Peter Kairouz, Xiao Chen, Lalitha Sankar, and Ram Rajagopal. 2017. Context-aware generative adversarial privacy. *Entropy* 19, 12 (2017), 656.

[12] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. (2001–). http://www.scipy.org/

[13] Kleomenis Katevas, Hamed Haddadi, and Laurissa Tokarchuk. 2014. Poster: Sensingkit: A multi-platform mobile sensing framework for large-scale experiments. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 375–378.

[14] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[15] Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361, 10 (1995), 1995.

[16] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. 2017. DEEProtect: Enabling Inference-based Access Control on Mobile Sensing Applications. *arXiv preprint arXiv:1702.06159* (2017).

[17] Chris YT Ma and David KY Yau. 2015. On information-theoretic measures for quantifying privacy protection of time-series data. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. ACM, 427–438.

[18] Sumit Majumder, Emad Aghayi, Moein Noferesti, Hamidreza Memarzadeh-Tehran, Tapas Mondal, Zhibo Pang, and M Deen. 2017. Smart Homes for Elderly Healthcare - Recent Advances and Research Challenges. *Sensors* 17, 11 (2017), 2496.

[19] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).

[20] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. 2018. Protecting sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*. ACM, 2.

[21] Mohammad Malekzadeh, Richard G Clegg, and Hamed Haddadi. 2018. Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis. In *Internet-of-Things Design and Implementation (IoTDI), 2018 IEEE/ACM Third International Conference on*. IEEE, 165–176.

[22] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*. Springer, 52–59.

[23] Natalia Neverova, Christian Wolf, Griffin Lacey, Lex Fridman, Deepak Chandra, Brandon Barbello, and Graham Taylor. 2016. Learning human identity from motion patterns. *IEEE Access* 4 (2016), 1810–1820.

[24] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Hamid R Rabiee. 2019. Deep Private-Feature Extraction. *IEEE Transactions on Knowledge and Data Engineering*.

[25] Borzoo Rassouli and Deniz Gündüz. 2018. Optimal Utility-Privacy Trade-off with the Total Variation Distance as the Privacy Measure. *arXiv preprint arXiv:1801.02505* (2018).

[26] Nisarg Raval, Ashwin Machanavajjhala, and Jerry Pan. 2019. Olympus: Sensor Privacy through Utility Aware Obfuscation. *Proceedings on Privacy Enhancing Technologies* 2019, 1 (2019), 5–25.

[27] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[28] Lalitha Sankar, S Raj Rajagopalan, and H Vincent Poor. 2013. Utility-privacy tradeoffs in databases: An information-theoretic approach. *IEEE Transactions on Information Forensics and Security* 8, 6 (2013), 838–852.

[29] Sandra Servia-Rodríguez, Liang Wang, Jianxin R Zhao, Richard Mortier, and Hamed Haddadi. 2018. Privacy-Preserving Personal Model Training. In *Internet-of-Things Design and Implementation (IoTDI), 2018 IEEE/ACM Third International Conference on*. IEEE, 153–164.

[30] Ali Shahin Shamsabadi, Hamed Haddadi, and Andrea Cavallaro. 2018. Distributed One-class Learning. In *IEEE International Conference on Image Processing (icip 18)*. IEEE.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[32] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. 2017. Privacy Loss in Apple's Implementation of Differential Privacy on macOS 10.12. *arXiv preprint arXiv:1709.02753* (2017).

[33] Apple Differential Privacy Team. 2017. Learning with privacy at scale. *Online at: https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html* (2017).

[34] Ardhendu Tripathy, Ye Wang, and Prakash Ishwar. 2017. Privacy-Preserving Adversarial Networks. *arXiv preprint arXiv:1712.07008* (2017).

[35] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, 1096–1103.

[36] Jun Wang, Shubo Liu, and Yongkai Li. 2015. A review of differential privacy in individual data release. *International Journal of Distributed Sensor Networks* 11, 10 (2015), 259682.

[37] Fengjun Xiao, Mingming Lu, Ying Zhao, Soumia Menasria, Dan Meng, Shangsheng Xie, Juncai Li, and Chengzhi Li. 2018. An information-aware visualization for privacy-preserving accelerometer data sharing. *Human-centric Computing and Information Sciences* 8, 1 (2018), 13.

[38] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. 2015. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.