Machine Learning Algorithms for Privacy-preserving Behavioral Data Analytics

by

Mohammad Malekzadeh

Bachelor in Computer Engineering 2008
Master in Information Technology Engineering 2011

A dissertation presented for the degree of
Doctor of Philosophy
in the subject of
Computer Science
submitted to

Queen Mary
University of London

School of Electronic Engineering and Computer Science
Queen Mary University of London
E1 4NS, London, United Kingdom

2020

Statement of Originality

I, Mohammad Malekzadeh, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated.
I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.
I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.
I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.
The copyright of this thesis rests with the author; no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature: Mohammad Malekzadeh

Date: 08-12-2020

Primary Supervisor
Prof. Andrea Cavallaro

Second Supervisor
Dr. Richard G. Clegg

Author
Mohammad Malekzadeh

Machine Learning Algorithms for Privacy-preserving Behavioral Data Analytics

Abstract

Behavioral patterns observed in data generated by mobile and wearable devices are used by many applications, such as wellness monitoring or service personalization. However, sensitive information may be inferred from these data when they are shared with cloud-based services. In this thesis, we propose machine learning algorithms for data transformations to allow the inference of information required for specific tasks while preventing the inference of privacy-sensitive information. Specifically, we focus on protecting the user's privacy when sharing motion-sensor data and web-browsing histories.

Firstly, for human activity recognition using data of wearable sensors, we introduce two algorithms for training deep neural networks to transform motion-sensor data, focusing on two objectives: (i) to prevent the inference of privacy-sensitive activities (*e.g.* smoking or drinking), and (ii) to protect user's sensitive attributes (*e.g.* gender) and prevent the re-identification of user. We show how to combine these two algorithms and propose a compound architecture that protects both sensitive activities and attributes. Alongside the algorithmic contributions, we published a motion-sensor dataset for human activity recognition.

Secondly, to prevent the identification of users using their web-browsing behavior, we introduce an algorithm for privacy-preserving collaborative training of contextual bandit algorithms. The proposed method improves the accuracy of personalized recommendation agents that run locally on the user's devices. We propose an encoding algorithm for the user's web-browsing data that preserves the required information for the personalization of the future contents while ensuring differential privacy for the participants in collaborative training.

In addition, for processing multivariate sensor data, we show how to make neural network architectures adaptive to dynamic sampling rate and sensor selection. This allows handling situations in human activity recognition where the dimensions of input data can be varied at inference time. Specifically, we introduce a customized pooling layer for neural networks and propose a customized training procedure to generalize over a large number of feasible data dimensions. Using the proposed architectural improvement, we show how to convert existing non-adaptive deep neural networks into an adaptive network while keeping the same classification accuracy.

We conclude this thesis by discussing open questions and the potential future directions for continuing research in this area.

Acknowledgments

First and foremost I would like to express my sincere gratitude to my supervisors: Prof. Andrea Cavallaro, Dr. Richard G. Clegg, and Dr. Hamed Haddadi. I could not have imagined having a better chance than the continuous mentorship, trust, inspiration, and support I have had from three brilliant teachers throughout the whole period of my Ph.D. studies. Particularly, I thank Andrea for his detailed comments and very helpful advice on each and every aspect of my Ph.D. research. I am grateful to Richard for his precise and thoughtful suggestions and his accurate notes after each meeting which was unique to me. I am thankful to Hamed for all amazing support and mentorship at the beginning of my Ph.D. study as well as the great experience and guidance throughout an internship at the Databox project and the fun and exciting moments I have had when I was visiting him at Imperial College London.

I was also lucky to meet many more wonderful people during the span of my Ph.D. research. I would like to thank my independent assessor, Prof. Pat Healey, for his help and comments during the various stages of my Ph.D. research. I am grateful to Dr. Benjamin Livshits and Dr. Dimitrios Athanasakis for their amazing mentorship during an industrial internship at Brave Research. I am also so grateful to Prof. Deniz Gunduz for his comments on my Ph.D. research and his wonderful help and support during writing my Ph.D. thesis. I thank Dr. Arman Khouzani, Dr. Mustafa Bozkurt, and Dr. Anthony Constantinou for the support and guidance provided during my experience as a teaching assistant. I sincerely thank my examiners, Prof. Miguel Rodrigues and Dr. Rik Sarkar, for reading my thesis carefully, for their constructive and insightful comments, and for useful discussion during the viva examination.

I thank my fellow lab mates and friends for their support, beneficial discussions, and all the fun we have had at Queen Mary Univesity of London, Imperial College London, and Brave Research. A special thank you to Ali, Alessio, Alex, Anastasia, Anna, Antonio, Ashish, Burak, Changjae, Daniel, David, Dimitrios, Elona, Gareth, Girmaw, Gianni, Ignacio, Inigo, Janice, Lida, Lin, Lucia, Minos, Mohamed, Nitish, Nikesh, Panos, Ranya, Roman, Ricardo, Riccardo, Shahanawaz, Soomi, Sophie, Stan, Vandana, Vincent, Yousef, Yuchen, and Yuting.

Last but not least, is *the family.* I would like to especially thank Ellie: thank you so much for your wonderful inspiration and happiness you brought to our lives, and for supporting me in every moment of this journey. I thank my mother for everything that she has done for me and for what I have and proud of right now. I thank my sisters, my brothers, my nieces, my nephews, and the rest of my family and friends back home, for their unconditional support, happiness, encouragement, and much more that could be written on these pages. Thank you all. The family has always been and will be *the reason*!

This thesis was supported by funding from the Life Sciences Institute at Queen Mary University of London.

Dedicated to all who believe in both humanitarianism and animal rights.

Contents

Abbreviations

AAE	Anonymizing AutoEncoder
Accl	Accelerometer
AE	AutoEncoder
App	(Software) Application
CBA	Contextual Bandit Algorithms
CTR	Click-Through Rate
DANA	Dimension Adaptive Neural Architecture
DAP	Dimension Adaptive Pooling
DAT	Dimension Adaptive Training
DNN	Deep Neural Network
DP	Differential Privacy
DTW	Dynamic Time Warping
ESA	Encode-Shuffle-Analyze
FNN	Feedforward Neural Network
GAN	Generative Adversarial Network
Gyro	Gyroscope
HAR	Human Activity Recognition
Hz	Hertz
IP	Informational Privacy
LDP	Local Differential Privacy
MSE	Mean Squared Error
P2B	Privacy-Preserving Bandits
RAE	Replacement AutoEncoder
RNN	Recurrent Neural Network
SSA	Singular Spectrum Analysis
SPP	Spatial Pyramid Pooling
t-SNE	t-distributed Stochastic Neighbor Embedding
UCB	Upper Confidence Bound

Notation

The notation that is used throughout this thesis:

General Notation

$\mathbf{a}$	A random variable
a	An unknown scalar (integer or real)
A	A known (constant) scalar
$\boldsymbol{a}$	A (column) vector
$\boldsymbol{A}$	A matrix
a_i	Element i of a vector
$a_{i,j}$	Element i, j (row, column) of a matrix
$\boldsymbol{a}^{\mathsf{T}}$	Transpose of a vector
$\mathbb{A}$	A set
$\mathbb{A}^i$	The i-th item in a set
$\mathcal{A}(\cdot)$	A function
$\mathcal{A}(\cdot;\theta)$	A function parameterized by a set of explicit parameters
$\texttt{size}(\mathbb{A})$	Cardinality of a set
$\lvert a \rvert$	Absolute value of a scalar
$[a, b)$	The real interval including a but not including b
$\triangleq$	Delta equivalent

Spacial Cases

$\texttt{E}(\mathbf{a})$	The expected value of a random variable
$\texttt{H}(\mathbf{a})$	Shannon entropy of a random variable
$\texttt{I}(\mathbf{a};\mathbf{b})$	Shannon mutual information between two random variables
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\mathbb{N}$	The set of natural numbers
$\texttt{P}(\mathbf{a})$	Probability distribution over a random variable
$\texttt{P}(\mathbf{a} \mid \mathbf{b})$	Conditional probability distribution over a random variable
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^{A \cdot B}$	A one-dimensional data, of size A product B
$\mathbb{R}^{A \times B}$	A two-dimensional data, of size A by B
$\{1, \dots, n\}$	The set of all integers between 1 and n

Chapter 1

Introduction

1.1 Motivation

The potentially *sensitive* information inferred from our behavioral data can be unwillingly used to manipulate our decisions [1, 2], affect our reputations [3], reveal our private activities [4], cause unfair discrimination against us [5], or provoke social embarrassment due to the leak of private information [6, 7]. Such behavioral data can be sold to insurance companies and digital marketers or even shared with courts of law or government organizations [8, 9]. If your *secret* data is leaked , *e.g.* your password, you can change it and make it secret again; but if your *private* data is leaked *e.g.* the name of your first pet which you may use as part of your password, then it becomes public and it never can become private again! Thus, safeguarding our personal data before sharing it with third-party applications (apps) is getting increasingly important, and motivated by the law [10, 11].

An individual's *data privacy* can be defined as "the right to select what personal information about me is known to what people" [12]. We believe, as the main motivation of this thesis, that the least we should have knowledge and control of is *the type and amount of information* that can be discovered from our data [13, 14, 15, 16]. To this end, we need *algorithms* that give us such *control* of our data, while allowing us to utilize cloud-based apps, such as health monitoring [17] or content personalization [18], that require having access to our data [19, 20, 21]. Despite their applicability to single-valued, or low-dimensional, static data, current privacy-preserving algorithms [22, 23, 24, 25, 26, 27] are not well-suited to multivariate data generated by mobile and wearable devices equipped with varying types of inertial sensors and third parties' apps; mainly because these algorithms (explained in Chapter 2) are designed for the extraction of population-level statistics from large datasets, and not the release of individual-level multivariate data.

We generate a considerable amount of data through our daily interactions with smart devices, or simply just by carrying them. For instance, wearing fitness trackers for monitoring personal health and wellness during the day, and even the night's sleep, is becoming pervasive [28]. Health-monitoring apps [29, 17] utilize data collected from motion sensors embedded in mobile and wearable devices to infer our body movements and temporal changes in our physiological state [30, 31]. Hence, sharing our sensor data helps to track our activities [32], interactions [33], mood [34, 35], and sleep quality [36, 37]. As another example, web browsing is an unavoidable daily routine, and we may welcome personalization of contents through collecting and analyzing a history of our past content browsing behavior to improve our future experiences [38, 39, 18]. A web browser can gather the history of the most visited websites to recommend personalized news articles that are attractive to us [40, 41], or a social media app can monitor our engagement in different contents to propose interesting pages or people to follow [42].

The main obstacle in utilizing such motivational apps is the risk of revealing potentially *sensitive information* that a *user* might wish to keep private. Sensor data can reveal sensitive habits such as smoking [43], attributes such as age and gender [44], or enable the re-identification of the user [45]. Continuous monitoring of ambient light [46] or accelerometer [47, 48] sensor can be used to extract sequences of entered text on smartphones and consequently the user's password. Recommendation apps, that keep track of the user's engagement in different types of contents, can model similarities across users that may reveal a user's sexual orientation, age group, approximate location, or political views which collectively can re-identify the user [49, 50, 51, 52, 53, 54].

Therefore, while we need to share our data with apps to obtain *utility*, we are also concerned about our *privacy*. A naive algorithm, which is often found to be used, is the extreme "all-or-nothing" that allows us to choose whether to grant an app permission for collecting data or not (*e.g.* permission over location). Such binary algorithms only allow us to choose one of two extremes: *perfect utility*, by giving them permission, or *perfect privacy*, by revoking the permission [55]. More flexible algorithms should allow us to grant permission over a controlled type and amount of information, customized according to the app's requirements. Hence, an important motivation is to build *privacy-preserving algorithms* that establish an acceptable utility-privacy trade-off between the protection of *sensitive* private information and the maintenance of *required* non-private information in the shared data, while our data can also contain some *neutral* information irrelevant to the target utility and not critical to our privacy.

1.2 Objectives

In this thesis, we design and evaluate machine learning algorithms that aim to provide *users* of cloud-assisted apps more control and protection over the type and amount of information that can be discovered from their data. Considering two specific, but pervasive, types of user-generated data, we evaluate the performance of the proposed algorithms in terms of the achieved *utility* and *privacy* trade-off; according to the defined criteria for the user and target app. First, we consider *motion-sensor streams* of mobile and wearable devices that are in the shape of multivariate time-series, and capture the user's gestures and activities with fine granularity. Second, we consider *browsing history* of web pages which are in the shape of temporal histograms that indicate the temporal interests of the user, and consequently may reveal their identity.

As an example, consider a smartwatch step-counter app that measures the user's body movement through motion sensors like an accelerometer. Static acceleration shows the magnitude and direction of the earth's gravitational force, and helps to recognize the user's posture; whereas dynamic acceleration shows changes in the motion velocity of the user that can be mapped to the user's activities [56]. We may specify the recognition of some user's activities as the *required* information for the utility of step-counting, such as *walking*, *jogging*, or *stair-stepping*. Also, there might be a set of *sensitive* activities that a user wish to keep private, such as *smoking* and *typing*, or *sensitive* inferences such as *gender*. Here, some activities may be neither *required* nor *sensitive* and be considered as the *neutral* activities, such as *sitting* or *standing*. We assume that such apps are *honest* in extracting their *required* information from shared data and in providing useful services to the users. But they may also *curiously* extract *sensitive* information that can be used to invade the user's privacy [57].

The focus of this thesis is on *privacy and utility preserving algorithms* for applying *data transformations* on the user's data before sharing them with curious cloud-assisted apps. Unlike previous approaches, to be discussed in Chapter 2, the proposed algorithms aim for applying online data transformation at the user side, rather than collecting all users' data and then applying a one-shot offline data transformation over the collected dataset. Moreover, previous works mostly considered single-valued static data, or they are more applicable for gathering population-level aggregated statistics [22, 23, 24, 25, 26, 27], while the algorithms proposed in this thesis aim for multivariate temporal data at the user-level.

1.3 Problem Definition

In sharing personal data with apps and cloud services, there is always the possibility of sharing more information than what is actually needed to be shared. In the last decades, numerous algorithms have been proposed for privacy-preserving data analysis: from *randomized response* [58] to *differential privacy* [59, 60], from *statistical confidentiality* [61] to *adversarial synthesis* [62], from *suppression and rounding* [63, 64] to *masking* [65] to *k-anonymity* [66] to *l-diversity* [67] to *crowd-blending* [68]. The common aspect among all these privacy-preserving algorithms, and their extensions and relaxations, is *data transformation*: all mechanisms, either deterministically or randomly, transform the original data into other representations, at the user side (*i.e.* local) or after collecting the data (*i.e.* central). However, there is no one-of-a-kind algorithm suitable for all data types, and, even for the same data type, the suitable algorithm can vary depending on the app's requirements. For example, the required information that the app needs to infer from data, the privacy metric that is used to quantify the privacy loss, the threat model assumptions on who does the user trust, and where the user's data will be processed.

In this thesis, we design privacy-preserving algorithms that aim to release a transformed version of the original data such that the transformed data is still useful for achieving the required utility, while revealing as minimum as possible information that is sensitive to the user's privacy. We consider the setting depicted in Figure 1.1 where we have two main parties called: the *user* (*e.g.* a smartwatch wearer) and the *app* (*e.g.* a software application and its corresponding cloud service provider). In each iteration of data sharing $i \in \mathbb{N}$, user generates data $\boldsymbol{X}^i \in \mathbb{R}^{w \times h}$ with dimensions ($w \in \mathbb{N}, h \in \mathbb{N}$). Specifically, we assume that $\boldsymbol{X}^i$ represents the captured data of a *time window*. For example, a 2.5-seconds time window of data generated by the accelerometer, gyroscope, and magnetometer of a smartwatch (each having three axes), with a 50 Hz sampling rate, has dimensions of $w = 125$ and $h = 9$, where each sample point shows a sensory reading. Or, a time window of the recent 100 web-page visits has dimensions of $w = 100$ and $h = 1$, where each sample point shows the category of the visited page. We also assume that each $\boldsymbol{X}^i$ is independent of other $\boldsymbol{X}^j$, for all $j \neq i$. Also, given the user, the data $\boldsymbol{X}^i$ is sampled from an identical distribution, for all i. For example, we assume that the motion characteristics of a user's *walking* activity do not change through time. Although the independence assumption is not always held in practice, the estimation of the non-linear dependency structure between a set of multivariate time windows is too difficult [69]; especially when we do not have good knowledge about the data generation process and the joint distribution $\mathtt{P}(\boldsymbol{X}^i, \boldsymbol{X}^j)$. For the sake of

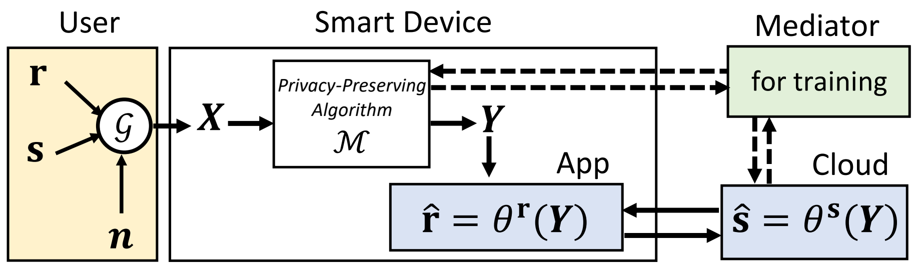

User
Smart Device
Mediator
for training
r
s
n
$\mathcal{G}$
X
Privacy-Preserving Algorithm $\mathcal{M}$
Y
App
Cloud
$\hat{\mathbf{r}} = \theta^{\mathbf{r}}(\boldsymbol{Y})$
$\hat{\mathbf{s}} = \theta^{\mathbf{s}}(\boldsymbol{Y})$

Figure 1.1: The general setting for temporal data sharing considered in this thesis. User is modelled as an unknown function $\mathcal{G}$. The observed data $\boldsymbol{X}$, captured by the device, is informative about three hidden random variables: *required* (non-private) $\mathbf{r}$, *sensitive* (private) $\mathbf{s}$, and *neutral* $\mathbf{n}$. A privacy-preserving algorithm, $\mathcal{M}$, transforms $\boldsymbol{X}$ into $\boldsymbol{Y}$ before sharing with an app. The app then computes $\hat{\mathbf{r}}$ and $\hat{\mathbf{s}}$ as the estimation of $\mathbf{r}$ and $\mathbf{s}$, using models empirical models parameterized by $\theta^{\mathbf{r}}$ and $\theta^{\mathbf{s}}$, respectively. We assume that except App and Cloud that are under the *app*'s control, other components are under the *user*'s control. Dashed arrows show interactions during the training of the algorithm; solid lines show data flow during the test time.

brevity, we remove subscript i when it is not necessary in the following.

In this setting, $\boldsymbol{X}$ is a multivariate *observed* variable that carries information about at least two *hidden* discrete variables: *required* non-private variables $\mathbf{r} \in \{1, 2, \cdots, R\}$ (*e.g.* user's activity or step counts), and *sensitive* private variables $\mathbf{s} \in \{1, 2, \cdots, S\}$ (*e.g.* user's gender or identity), where R and S denote the domain size of $\mathbf{r}$ and $\mathbf{s}$, respectively. Beside these two, $\boldsymbol{X}$ might naturally carry information about some other hidden variables which we call them *neutral* variables $\mathbf{n}$ (*e.g.* on-body position of the mobile device). The temporal value of hidden variables are not directly accessible and have to be inferred from temporal data $\boldsymbol{X}$. Let a data transformation algorithm $\mathcal{M}$ take $\boldsymbol{X}$ and produce data $\boldsymbol{Y} = \mathcal{M}(\boldsymbol{X})$ for sharing with the app. The dimensions of $\boldsymbol{Y}$ can be the same as the dimensions of $\boldsymbol{X}$ or not, depending on the context.

Given the received $\boldsymbol{Y}$ an empirical machine learning model, parameterized by $\theta^{\mathbf{r}}$, computes the $\hat{\mathbf{r}} = \theta^{\mathbf{r}}(\boldsymbol{Y})$ that denotes the estimation of $\mathbf{r}$. Let us assume that $\theta^{\mathbf{r}}$ (as a potential app's model) *honestly* estimates $\mathbf{r}$, to provide the desired utility to its user. For example, an activity monitoring app can provide its user some daily statistics about number of steps or the amount of calories they have burnt. This honesty is important, as otherwise the app cannot attract any user for its offered services. But this app may also *curiously* use a model parameterized by $\theta^{\mathbf{s}}$ to discover the value of $\mathbf{s}$ by computing $\hat{\mathbf{s}} = \theta^{\mathbf{s}}(\boldsymbol{Y})$, when the app get access to and collect the user's data.. This is usually called a *honest-but-curious* threat model [70], which in this thesis it particularly means that while users get the promised service from the app's provider (*i.e.* honesty), users cannot make sure that their data is not used for other purposes (*i.e.* curiosity).

Let data utility $u_{\boldsymbol{Y}} \in [0,1]$, over N rounds of data sharing, be defined as the average of correct prediction of $\mathbf{r}$ by the app:

$$u_{\boldsymbol{Y}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\Big[\theta^{\mathbf{r}}(\boldsymbol{Y}^i) = \mathbf{r}^i\Big], \tag{1.3.1}$$

where $\mathbf{1}[\cdot]$ denotes the indicator function that outputs 1 if its condition holds, otherwise 0. In this thesis, in Chapter 3 and 4 where we consider multi-class classification tasks, $u_{\boldsymbol{Y}}$ is interpreted as the *classification accuracy* of the model $\theta^{\mathbf{r}}$, and in Chapter 5 where we consider recommendation tasks, $u_{\boldsymbol{Y}}$ is interpreted as the *average reward* of the model $\theta^{\mathbf{r}}$.

Let privacy cost $c_{\boldsymbol{Y}} \in [0,1]$, over N rounds of data sharing, be defined as the absolute difference between the average of correct prediction of $\mathbf{s}$ by the app and the accuracy of the app's prior belief over $\mathbf{s}$, $\mathtt{G}(\mathbf{s})$:

$$c_{\boldsymbol{Y}} = \left|\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\Big[\theta^{\mathbf{s}}(\boldsymbol{Y}^i) = \mathbf{s}^i\Big] - \mathtt{G}(\mathbf{s})\right|. \tag{1.3.2}$$

In other words, we assume that $c_{\boldsymbol{Y}}$ measures the changes in the app's posterior belief after observing $\boldsymbol{Y}$.

It is worth noting that the optimal value of $u_{\boldsymbol{Y}}$ is always 1, while the optimal value of $c_{\boldsymbol{Y}}$ is 0; considering that any changes in the app's posterior belief on $\mathbf{s}$ can be interpreted as information leakage [71]. For example, in Chapter 4 where we consider *gender* as the sensitive attribute and we assume $\mathtt{G}(\mathbf{s}) = 0.5$, the optimal privacy is achieved when the average of correct prediction of $\mathbf{s}$ by the app is 50%. In Chapter 3, we consider performing some activities, *e.g. smoking*, as the sensitive behavior and we assume that the app's prior belief is that the user is not performing any of these activities (*i.e.* $\mathtt{G}(\mathbf{s}) = 0$). Thus, any correct predictions of such sensitive activities will cause a privacy cost. In Chapter 5, we consider that an app's prior belief on the presence or absence of the user in the shared dataset is 0.5., thus leaking any information about either the presence or absence of the user is considered as privacy cost.

Let a *privacy-preserving algorithm* be a function $\mathcal{M} : \boldsymbol{X} \to \boldsymbol{Y}$ that minimizes $c_{\boldsymbol{Y}}$ subject to maximizing $u_{\boldsymbol{Y}}$. Note that we always have $u_{\boldsymbol{Y}} \leq u_{\boldsymbol{X}}$ and $c_{\boldsymbol{Y}} \leq c_{\boldsymbol{X}}$. The reason can be found in the underlying structure $(\mathbf{r}, \mathbf{s}, \mathbf{n}) \to \boldsymbol{X} \to \boldsymbol{Y}$ constructed from our assumptions (see Figure 1.1), thus based on the data processing inequality [72], the maximum amount of information about $\mathbf{r}$ and $\mathbf{s}$ that can be inferred from the released data $\boldsymbol{Y}$ is equal to what one can infer from the original data $\boldsymbol{X}$.

We assume that the app only has access to the output of $\mathcal{M}$ at the test time (*i.e.* inference time), but it has complete knowledge of the $\mathcal{M}$ which models a strong adversary. For finding the appropriate algorithm $\mathcal{M}$ (*i.e.* training), we sometimes need help from a *mediator* to mediate between the user and a service provider. However, the proposed algorithms do not need such a mediator at test time on the user's side. The mediator in Chapter 3 and Chapter 4 is a trusted party that trains $\mathcal{M}$ on a (public or private) dataset and broadcasts the trained model back to the users. Similarly, in Chapter 5 we assume that the mediator is a trusted third party that anonymizes the communications between users and the app's server during training.

Finally, the dimensions, w and h, of the transformed data, $\boldsymbol{Y}$, may not always be the same as its input, $\boldsymbol{X}$. For example, due to privacy protection [4, 73, 74], energy preservation [75, 76, 77], or fault tolerance [78, 79] requirements, an algorithm may dynamically select/deselect some sensors or increase/decrease the sampling rate of the selected sensors, before sharing data with the app. Therefore, as an appendix to this thesis, we also look into solutions for making an app adaptive to such algorithms that may share variable-sized data with the app, in different situations, while still providing a reliable classification of the shared data. The motivation is that empowering apps with such capability will also facilitate designing more flexible privacy-preserving algorithms, because it allows apps to still recognize user's activity (with some level of accuracy) even when users decide to share data of which sensors and with what sampling rate.

The definitions that are used throughout this thesis are summarized in Table 1.1.

1.4 Thesis Contributions

The novelties and main contributions of this thesis are as follows.

First, we present the *Replacement AutoEncoder* (RAE): a deep neural network architecture that takes a time window of multivariate motion sensors, $\boldsymbol{X}$, as input and if $\boldsymbol{X}$ is generated through a user's *sensitive* activity, then the RAE transforms $\boldsymbol{X}$ into a time window $\boldsymbol{Y}$ such that the transformed data resembles time windows that are generated through a user's *neutral* activity. Otherwise, if $\boldsymbol{X}$ is generated through a user's non-sensitive (*i.e. required* or *neutral*) activity, the RAE produces a $\boldsymbol{Y}$ as similar as possible to $\boldsymbol{X}$; ideally $\boldsymbol{Y} = \boldsymbol{X}$. Although previous methods, such as filtering [80, 81, 82] or perturbation [83, 26], prevent the recognition of the exact sensitive activity, they have this major weakness of revealing that an unknown-but-sensitive activity has occurred. Our RAE covers this weakness as it not only eliminates the possibility of recognizing the user's sensitive activities, it also limits the possibility of detecting the occurrence of them. Moreover, unlike offline transformation approaches [84] that assume data

Table 1.1: Summary of the definitions used throughout this thesis.

Term	Description
activity	the daily activity of a wearer/user of a wearable/mobile device inferred from motion sensors
anonymization	preventing an app from identifying a user, among a set of potential candidates.
app	a software application installed on the *user*'s device and backed by a service provider
browsing history	a histogram of the type or category of visited contents by the *user*
behavioral data	information inferred from the *user*'s (*motion-sensors* or *browsing history*) data
curious app	an *app* that tries to infer the *user*'s *sensitive information*
context	the current state of a user's engagements inferred from their browsing history
data transformation	converting a sample data into another representation (of the same or different dimensions)
honest app	an *app* that provide the promised services to the *user*
identification	identifying a specific user among a set of potential candidates.
motion sensors	accelerometer, gyroscope, and magnetometer sensors embedded in mobile and wearable devices
neutral information	information that is neither *sensitive* nor *required*
required information	information that the *user* wants to share with the *app*
sampling rate	the number of data points captured in the predefined unit of time
sensitive information	information that the *user* wishes to keep private
sensor data dimensions	number of sensory streams, h, and the length of each stream, w.
time window	a bounded time interval which is characterized by a *length* and a *step-size*
user	the person who owns or interacts with mobile or wearable devices

is already collected, RAE is a trained model that can be used in an online setting.

Second, we propose the *Anonymization AutoEncoders* (AAE): a deep neural network architecture, trained through an adversarial training procedure using a customized objective function, to avoid the unexpected inference of the user's *sensitive* attributes or the re-identification of the user. The AAE takes sensor data over a time window $\boldsymbol{X}$, generated through a user's activity, and produces a time window $\boldsymbol{Y}$ which is informative about the user's activity but uninformative about the user's *sensitive* attribute $\mathbf{s}$ (*e.g.* gender or identity). Unlike other approaches [85, 86, 87, 88, 89], AAE regulates both encoder and decoder to shape the final output independently of the specific users in the training set. Moreover, we show how to cascade RAE and AAE into a compound architecture to build a unified framework that can protect both *sensitive activities* and a *sensitive attribute*.

Third, we introduce *Privacy-Preserving Bandits* (P2B): a system for collaborative and private training of contextual-based algorithms. P2B transform the histogram of the browsing behavior into a cluster code which is informative about the *required* information for the personalization of the contents while providing the user differential privacy guarantee when sharing their data for training a global model. Compared to other collaborative learning approaches [90, 91, 92], P2B is substantially less complicated as it does not need users to train a model locally and it does not need multiple rounds of communication between users and the server. Moreover, other privacy-preserving bandits algorithms [93, 94] consider a centralized setting where the server knows everything about the user except their responses to the recommended contents, but P2B is built upon a distributed setting where all the user's data are processed locally.

Forth, alongside the above algorithmic contributions, we collected and published *MotionSense* [95]: a dataset containing data of two sensors embedded in mobile devices: accelerometer and gyroscope. MotionSense includes 24 users of different gender, ages, weights, and heights, where each user performs 6 activities in 15 different trials. Thus, covering the shortcoming of other datasets [96, 97, 98, 99, 100] that only report the performed activities, and no information about the users who performed those activities. We use MotionSense for evaluating some of the proposed algorithms in this thesis, and by the time of writing this thesis, it also has been used by several other researchers to evaluate their methods.

Finally, as an appendix, we propose *Dimension-Adaptive Neural Architecture* (DANA) that includes a *dimension-adaptive pooling* layer to make deep architectures adaptive to temporal changes in sampling rate and in sensor availability, and a *dimension-adaptive training* procedure to generalize over the entire space of feasible data dimensions at the inference time. DANA allows us to transform existing non-adaptive deep architectures [101, 102, 103] into an adaptive model without the need for adding or removing their trainable parameters. While previous works address either sensor selection [96, 104, 105, 106, 107, 108] or adaptive sampling rate [109, 110, 111, 112, 113], DANA provides a unified solution to both requirements.

1.5 Associated Publications

Parts of the work detailed in this thesis have been presented in the following **J**ouranl, **C**onference, and **W**orkshop publications as well as **P**reprints in chronological order:

1. [C1] M. Malekzadeh, R. G. Clegg, H. Haddadi. "Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis." IEEE/ACM International Conference on Internet-of-Things Design and Implementation (IoTDI'18) pp. 165-176, Orlando, Florida, April 2018. (Chapter 3)

2. [W1] M. Malekzadeh, R. G. Clegg, A. Cavallaro, H. Haddadi. "Protecting sensory data against sensitive inferences." First EuroSys Workshop on Privacy by Design in Distributed Systems (W-P2DS), pp. 1-6, Porto, Portugal, 2018.(Chapter 4)

3. [C2] M. Malekzadeh, R. G. Clegg, A. Cavallaro, H. Haddadi. "Mobile sensor data anonymization." IEEE/ACM International Conference on Internet-of-Things Design and Implementation (IoTDI'19), pp. 49-58, Montral, Canada, April 2019. (Chapter 4)

4. [J1] M. Malekzadeh, R. G. Clegg, A. Cavallaro, H. Haddadi, "Privacy and utility preserving sensor-data transformations", In Journal of Pervasive and Mobile Computing, Elsevier, Volume 63, Article 101132, 2020. (Chapter 3 and 4)

5. [C3] M. Malekzadeh, D. Athanasakis, H. Haddadi, B. Livshits. "Privacy-Preserving Bandits." In Proceedings of Third Conference on Machine Learning and Systems (MLSys'20), Austin, Texas, March 2020. (Chapter 5)

6. [P1] M. Malekzadeh, R. G. Clegg, A. Cavallaro, H. Haddadi, "Dimension Adaptive Neural Architectures for Temporal Data." Under review. (Appendix A)

1.6 Organization of the Thesis

This thesis is organized as follows:

Chapter 2, Background and Related Work. We provide a literature review on privacy-persevering methods for sharing multivariate data, discuss the characteristics of the behavioral data that are of interest in this thesis, and present prerequisites that are needed for understanding the following chapters of this thesis. We also describe our contributed MotionSense dataset as well as other datasets that are used in this thesis.

Chapter 3, Sensitive Data Replacement. we consider the user's *sensitive* activities that can be unexpectedly discovered by a *human activity recognition* (HAR) app. We assume that the user shares a time window of multivariate motion sensors $\boldsymbol{X}$, generated by wearable devices while performing daily activities. The user wishes to prevent the app from inferring their *sensitive* activities $\mathbf{s}$, while allowing the inference of required activities $\mathbf{r}$, as accurate as possible. As a solution, we present the *Replacement AutoEncoder* (RAE): a deep neural network architecture that takes $\boldsymbol{X}$ as input and if $\boldsymbol{X}$ is generated through a user's *sensitive* activity, then the RAE produces a synthetic time window $\boldsymbol{Y}$ which aims to be similar to data generated through a user's *neutral* activity $\mathbf{n}$. Otherwise, if $\boldsymbol{X}$ is generated through a user's non-sensitive (*i.e. required* or *neutral*) activity, the RAE outputs a $\boldsymbol{Y}$ as similar as possible to $\boldsymbol{X}$; ideally $\boldsymbol{Y} = \boldsymbol{X}$. This efficiency is achieved by defining a procedure for training deep autoencoders that helps them approximating a two-part piece-wise function, such that the RAE learns to act as a transformation function if data is sensitive, otherwise act as the *identity function* if data is non-sensitive or required. We also design an attack on the RAE using a generative adversarial network (GAN) [114] to show that, even in a worst-case scenario where the app can have access to some user's original data, the RAE provides some level of privacy protection. This chapter is based on [C1] and parts of [J1].

Chapter 4, Sensor Data Anonymization. We consider HAR again, and discuss that even the user's non-sensitive activities can be exploited by a curious app to infer the user's *sensitive* attributes or enable the re-identification of the user. To avoid such unexpected discovery, we propose the *Anonymization AutoEncoders* (AAE): a deep neural network architecture trained through an adversarial training procedure using a customized objective function. The AAE takes a sensor time window $\boldsymbol{X}$, generated through a user's activity, and produces a time window $\boldsymbol{Y}$ which is informative about the user's activity $\mathbf{r}$, but uninformative about the user's *sensitive* attribute $\mathbf{s}$ (*e.g.* gender or identity). Hence, while HAR apps can recognize what activity the user is performing, it becomes much more difficult for them to infer who is performing that activity. We propose a multi-objective loss function for training AAE that helps to minimize the *privacy loss* $c_{\boldsymbol{Y}}$ by minimizing the mutual information between *sensitive* variable $\mathbf{s}$ and the shared data $\mathbf{Y}$. The AAE is designed to be flexible such that its comprising component and the multi-objective loss function can be flexibly tuned for a desirable trade-off between privacy and utility. Moreover, we show how to cascade RAE and AAE into a compound architecture to build a unified framework that can protect both *sensitive activities* and a *sensitive attribute*. This chapter is based on [W1], [C2], and parts of [J1].

Chapter 5, Privacy-Preserving Contextual Bandits. We consider content provider apps (*e.g.* social media or web browsers) and privacy concerns related to the identification of users by making a profile of their potentially *sensitive* interests for *personalization* purposes. We introduce *Privacy-Preserving Bandits* (P2B): a system for collaborative and private training of contextual-based algorithms [41, 115]. The P2B lets the app that is running locally on users' devices to contribute useful feedback to other apps through centralized model updates in a differentially-private manner. We show how to transform the histogram of the browsing behavior $\boldsymbol{X}$ into a cluster code y which is informative about the *required* information for the personalization of the contents while providing the user differential privacy guarantee when sharing y for training a global model. We compare P2B with a non-private system (*i.e.* centralized training without privacy guarantee) and a fully-private system (*i.e.* only local training), and show the performance on both synthetic benchmarks and real-world data, and discuss why P2B can be an effective solution to account for the challenges arising in on-device privacy-preserving personalization. This chapter is based on publication [C3], and has been carried out during an industrial placement at Brave Research, under the co-supervision of Dr. Dimitrios Athanasakis and Dr. Benjamin Livshits.

Chapter 6, Conclusion and Future Work. We summarize the findings presented in the previous chapters and discuss the directions for future research that may address existing open questions in privacy-preserving behavioral data analytics that are not covered in this thesis.

Appendix A, Dimension-Adaptive Neural Architecture. We discuss that sensors embedded in wearable and mobile devices allow for dynamic configuration and customization of sensor streams and sampling rates, depending on the privacy, accuracy, and power consumption requirements of each app. We show that changes in the dimensions w and h of the input data can cause considerable accuracy loss and even application failures, or enforce data pre-processing or unnecessary computations by the neural network. Since current deep neural network architectures for multivariate sensor data $\boldsymbol{X} \in \mathbb{R}^{w \times h}$ are mostly designed for data coming from a fixed set of sensors, with a fixed sampling rate [116, 101, 103, 102, 117, 118, 119]. To address this problem, we propose *Dimension-Adaptive Neural Architecture* (DANA). We introduce a *dimension-adaptive pooling* layer that makes deep architectures robust to temporal changes in sampling rate and in sensor availability, then building on this architectural improvement, we present a *dimension-adaptive training* procedure to generalize over the entire space of feasible data dimensions at the inference time. DANA is a unified framework that transforms existing non-adaptive deep architectures into an adaptive model without the need for adding or removing their trainable parameters. We show that how a HAR app can provide a good classification accuracy to the user in dynamic situations where sensor data dimensions can change, accidentally (*e.g.* due to hardware failure such as missing a sensor) or deliberately (*e.g.* due to power saving strategies). This appendix is based on based on [P1].

Chapter 2

Background and Related Work

In this chapter, we discuss the nature and characteristics of the data types that we consider in this thesis, and provide the mathematical definition of the related privacy and utility metrics that we refer to in this thesis. We also elaborate on the related works in the literature of privacy-preserving data analytics and compare them in the context of user-level behavioral data sharing.

2.1 Time Series

A (real-valued or categorical) *sample datum* represents the state of a variable in time. An ordered set of samples, generated by the same source through time, shapes a *time series* [120]. Usually, a time series is captured regularly, based on a predefined *sampling rate*; known as equally spaced time series, such as the linear acceleration of the user's body recorded by a smartwatch's accelerometer in every 50 milliseconds. A time series can also be captured irregularly, based on an *event* happening, known as unequally spaced time series, for instance, type of the web-content that user is looking for, recorded by every page view in a browser app.

The *granularity* of an (equally spaced) time series describes the period between two successive data points. The *sampling rate* of a time series is the number of data points captured in the predefined unit of time (*e.g.* every second). For instance, a time series with a 20 milliseconds granularity produces 50 temporal data points per second, 50 Hertz (Hz). Time series can be *univariate*, including only one variable, or *multivariate*, containing two or more variables. If the statistical properties (such as mean, variance, and auto-correlation) of a time series do not change over time, it is called *stationary*, while in a *non-stationary* time series some statistical properties may change over time.

Correlation is an important aspect in time series that is utilized by predic-

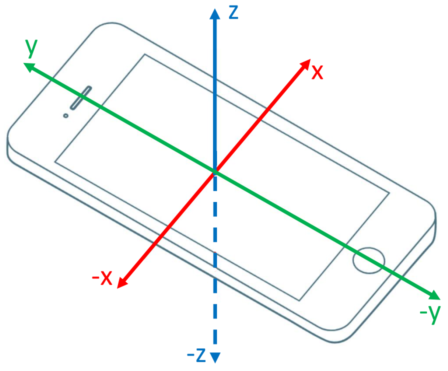


Figure 2.1: Every reading for a motion sensor (accelerometer, gyroscope, or magnetometer) includes three elements x, y , and z, representing the direction of the device motion [122].

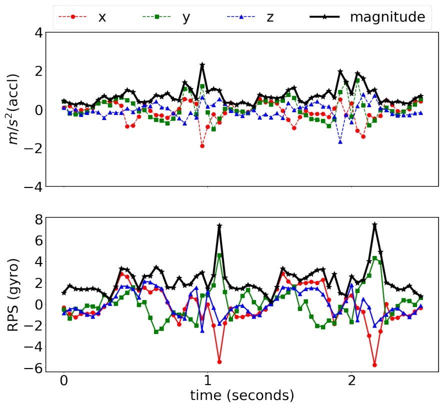


Figure 2.2: A time window of accelerometer (top) and gyroscope (bottom) sensor data for *walking* activity. RPS is revolutions per second, and m/s^2 is meters per second squared.

tion, classification, or clustering algorithms. Correlation, *e.g.* Pearson product-moment correlation [121], measures the strength and direction of linear relationships between pairs of real-valued variables. *Auto-correlation* is the correlation between two data points, x_t and x_{t-k}, of the same time series $\boldsymbol{x}$, where t is the current time and $k > 0$. *Cross-correlation*, on the other hand, is the correlation among two data points, x_t^1 and x_t^2, of two different time series, or two different variables of the same multivariate time series, $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ [120].

2.1.1 Motion Sensors

Personal devices, such as smartphones and smartwatches, are equipped with a collection of onboard motion sensors; such as accelerometers, gyroscopes, and magnetometers. These sensors repeatedly measure the value of a variable over time and across three-dimensional space (see Figure 2.1), producing time series that can be used to infer the user's fine-grained body movements or temporal changes in the user's physiological state [17]. These sensors facilitate many applications for users' health and wellness as they can help to infer the user's gestures, activities, and other behaviors [34]. For example, Figure 2.2 shows a 2.5 seconds time window of a 50 Hz time series collected from accelerometer and gyroscope sensors of a smartphone in the user's front pocket while walking. Each accelerometer's data point includes three numerical values representing the rate of change in the linear velocity across three-dimensional space (x, y, z). Similarly, values of a gyroscope's data point represent the angular velocity at which the device is rotating in the three-dimensional space.

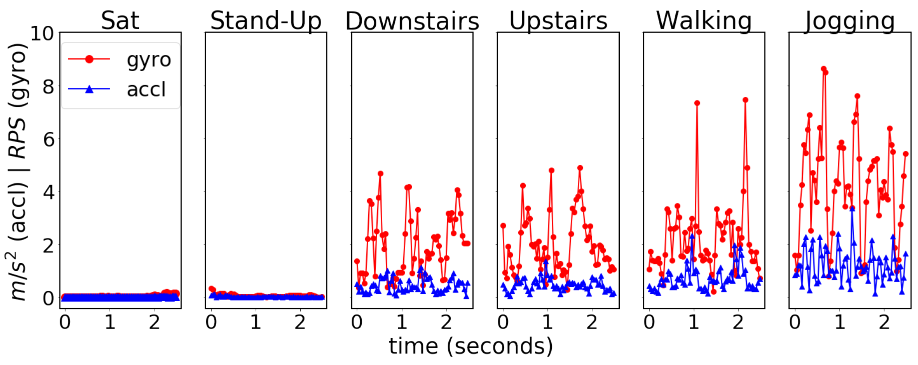


Figure 2.3: A time window of accelerometer (accl) and gyroscope (gyro) data for six different activities of a user.

Figure 2.2 also shows that the magnitude value ($\sqrt{x^2 + y^2 + z^2}$) of both sensors almost follow each other, especially for the peaks and periodicity, intuitively showing a strong correlation between two sensors. However, the correlation between two axes of the same sensor is less obvious, at least by the human eye. The distance between two peaks shows the stride length of the user that is a bit more than one second in this example. Figure 2.3 compares the magnitude values of the data from two sensors when the user performs six different activities. Note that motionless activities such as *sat* (*i.e.* sitting still) and *stand-up* (*i.e.* standing still) are difficult to be distinguished by the magnitude of sensors, as the device does not move significantly during these activities. Moreover, distinguishing among activities such as going *downstairs* or *upstairs* or *walking* (on a flat area) might not be trivial as the patterns in the data of these activities are quite similar.

2.1.2 Sampling Rate

Sampling rate has an important impact on the recognition of users' activity via mobile and wearable devices [110]. The Nyquist-Shannon *sampling theorem* [123, 124] specifies the sufficient sampling rate that allows us to completely reconstruct the original continuous-time signal from a received discrete-time signal: If a continuous-time data source contains no frequency components higher than H Hertz, then it is completely determined by giving a time series of sample points that are equally spaced $1/(2H)$ seconds apart. This states that every time series with a sampling rate equal to or more than the sufficient rate captures all the information included in the original, finite bandwidth signal.

Because of implicit uncertainties in human behavior, it is not straightforward to easily use the Nyquist-Shannon theorem [125] and define a minimum sampling rate that captures the frequency components necessary to discern dif-

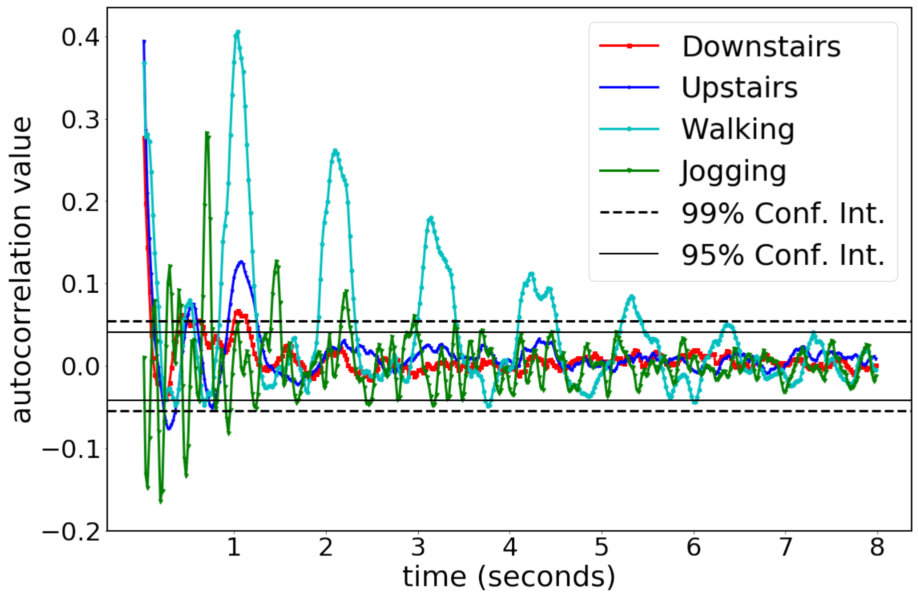


Figure 2.4: Auto-correlation of accelerometer data for four activities averaged over 24 users in MotionSense [95]. For example, samples of walking activity are correlated in a time window of about 5 seconds, while for jogging the correlations last for about 3 seconds. Correlation values outside the lines of the confidence interval (Conf. Int.) are considered as a statistically significant correlation.

ferent types of user's behaviors [110]. For example, accelerometers provide two types of measurements, namely static and dynamic acceleration [56]. Static acceleration is measured with respect to the earth's gravitational field and is useful for recognizing changes in the user's posture. For example, the direction of the gravitational force in a smartphone's accelerometer data can determine whether the phone is held vertically or horizontally, and the phone's position can be mapped to a specific user's posture. Dynamic acceleration measures changes in the velocity of the user's body and generates higher frequency patterns, depending on the user's body size. In lower sampling rates, signal aliasing happens to the acceleration patterns and activities that are characterized by the high frequencies in the data, become indiscernible from each other [112, 113].

The sufficient sampling rate not only varies depending on the set of activity classes or the position of the sensor on the user's body, but it even varies across different users while all other parameters are fixed. Khan *et al.* [110] show that the sufficient sampling rate across different datasets varies in the range 22Hz to 63Hz, for a 99% Kolmogorov-Smirnov similarity test [126], which is a non-parametric test for the equality of continuous probability distributions. Thus, the estimated probability distributions of corresponding time windows for the sufficient sampling rate against the original sampling rate of the dataset should

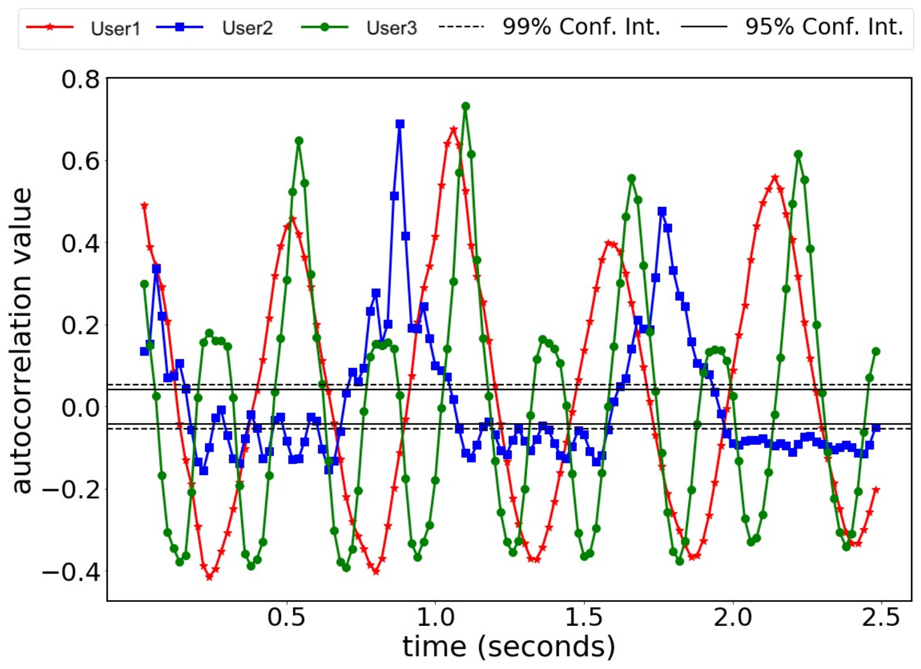


Figure 2.5: Auto-correlation of the accelerometer data for three different users while walking. The differences in correlation patterns between users can be due to several factors such as weight, stride length, walking speed, and even the tightness of the trouser.

become statistically similar, as measured by the Kolmogorov-Smirnov test.

2.1.3 Window-Based Classification

A typical practice in processing sensor data is to use a sliding time window to segment the incoming sensor streams into smaller tractable time periods. A sliding *time window* is a bounded time interval which is characterized by a *length* and a *step-size*, both in the unit of seconds. For example, to continuously process the latest five seconds (*i.e.* the length) of sensor data every second (*i.e.* the step), we need a sliding time window of the length 5 seconds with the step-size of 1 second.

Mostly, the aim of processing sensor data is to assign each time window to one of the predefined classes. There are different approaches, from classical machine learning algorithms, such as decision trees or support vector machines [127], to deep learning methods, such as convolutional or recurrent neural networks [128], for the discovery and learning of discerning patterns in sensor data. For all these algorithms, the length of the time window has an important effect on the classification accuracy [129, 130]. Short time windows accelerate the classification task, but they may negatively impact accuracy; while long time windows increase the computational complexity and may contain patterns related to two or more classes. Although, in real-world situations a user may perform some parallel activities at the same time, *e.g.* smoking while standing and typing an email on the phone, we do not yet have access to such a public dataset including parallel user activities. Thus, in our experiment throughout this thesis, each time window only contains one and only one user's activity, which is a common assumption in most previous related works [97, 99, 102, 103, 116, 117].

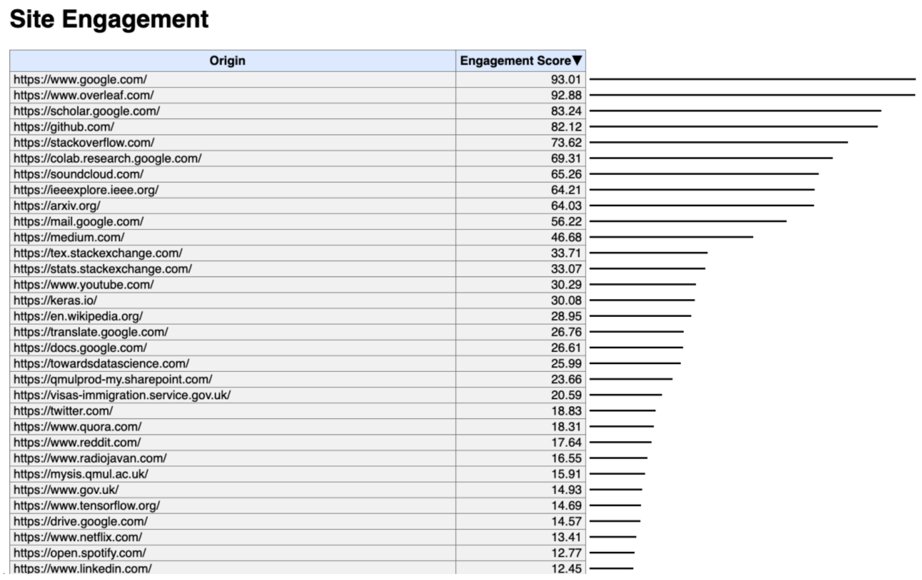
Figure 2.4 shows the average auto-correlation, over 45 seconds of data for 24 users, at varying time lags for the magnitude of accelerometer data for different activities. Note that each activity has a different period. Walking has the highest correlation, followed by Jogging, Upstairs, and Downstairs. The distance between two peaks can be connected to the stride length. There are also strong auto-correlations among samples inside a 2-second time window, whereas auto-correlations go under the confidence interval after about 5 seconds.

Figure 2.5 shows the auto-correlations of the same activity (walking) performed by three users. For different users, the intervals between two peaks are different, which shows the walking pace varies among different users. This user-identifiable pattern is a challenging feature to obscure before sharing the data. In fact, such different patterns in sensor data are what an app can exploit for user re-identification or to discover the user's gender.

2.1.4 Browsing History

Web browsing apps, for content-based advertisement [131], usually classify web contents into different categories such as educational, financial, sport, family, and entertainment. When a user is browsing the contents of a web page, the user's *browsing data* is updated to capture the recent user's interactions. Through time, the browsing history of a user can be used to identify the user's interests or temporal requirements. Hence, this data is a valuable source of information for personalization purposes [41].

A typical representation for the browsing history is obtained via *histograms*. For example, Figure 2.6, shows a histogram of my Site Engagement data [132]. Chromium browser keeps more than 4000 different histograms each representing accumulated data of a specific variable and capturing data from browser startup to current page load [133]. A histogram of browsing history is an indicator of the user's personal interests and needs [134], and can be used to re-identify the user. From Figure 2.6, one can easily guess that data is linked to a student of Queen Mary University of London, who is a heavy user of LaTeX, using deep learning libraries, reading about visa immigration rules, and listening to Persian



Site Engagement

Origin	Engagement Score▼
https://www.google.com/	93.01
https://www.overleaf.com/	92.88
https://scholar.google.com/	83.24
https://github.com/	82.12
https://stackoverflow.com/	73.62
https://colab.research.google.com/	69.31
https://soundcloud.com/	65.26
https://ieeexplore.ieee.org/	64.21
https://arxiv.org/	64.03
https://mail.google.com/	56.22
https://medium.com/	46.68
https://tex.stackexchange.com/	33.71
https://stats.stackexchange.com/	33.07
https://www.youtube.com/	30.29
https://keras.io/	30.08
https://en.wikipedia.org/	28.95
https://translate.google.com/	26.76
https://docs.google.com/	26.61
https://towardsdatascience.com/	25.99
https://qmulprod-my.sharepoint.com/	23.66
https://visas-immigration.service.gov.uk/	20.59
https://twitter.com/	18.83
https://www.quora.com/	18.31
https://www.reddit.com/	17.64
https://www.radiojavan.com/	16.55
https://mysis.qmul.ac.uk/	15.91
https://www.gov.uk/	14.93
https://www.tensorflow.org/	14.69
https://drive.google.com/	14.57
https://www.netflix.com/	13.41
https://open.spotify.com/	12.77
https://www.linkedin.com/	12.45

Figure 2.6: Screenshot of Site Engagement for data captured by my personal Chromoium browser at 25-08-2020. The engagement score (range between 0 and 100) of an Origin (*i.e.* a URL) is a cumulative function of how long the user scrolls, how many clicks are made, and how many character are typed during each visit [132].

music!

2.2 Privacy Definitions

In this thesis, we focus on two major categories of privacy algorithms known as *differential privacy* (DP) [60] and *inferential privacy* (IP) [71]

2.2.1 Differential Privacy

Intuitively, a DP algorithm bounds the ability of attackers to distinguish whether or not the data of a specific user, $\boldsymbol{x}$, is used in the computation; holding the participation of all other user's data fixed assuming a threat model where all-but-one can collude. This is known as the worst-case adversary of DP where we assume that an adversary can have complete knowledge about all the user's in the dataset, except one of them who is the target of the attack.

Definition 2.1 (**Differential Privacy** [59, 60])**.** Given $\epsilon, \delta \geq 0$, a data sharing algorithm $\mathcal{M}$ satisfies $(\epsilon, \delta)-$differential privacy if for all pairs of neighbor datasets $\mathbb{X}$ and $\mathbb{X}'$ differing in only one sample $\boldsymbol{x}$—such that $\mathbb{X} = \mathbb{X}' \cup \{\boldsymbol{x}\}$ or

$\mathbb{X}' = \mathbb{X} \cup \{\boldsymbol{x}\}$—and for all $\mathbb{Y} \subset \mathit{Range}(\mathcal{M})$, we have

$$\mathtt{P}\Big(\mathcal{M}(\mathbb{X}) \in \mathbb{Y}\Big) \leq e^{\epsilon}\mathtt{P}\Big(\mathcal{M}(\mathbb{X}') \in \mathbb{Y}\Big) + \delta, \tag{2.2.1}$$

where ϵ is the parameter that specifies the privacy loss, δ is considered as the "probability of failure" in ensuring that the inequality holds, and the probability distribution $\mathtt{P}$ is over the internal randomness of the algorithm $\mathcal{M}$, holding the dataset fixed.

An algorithm with $\delta = 0$ is called a pure DP because it never fails to bound the privacy loss. An algorithm with $\delta > 0$ is called approximate DP because it approximately ensures, with probability $1 - \delta$, that the privacy loss is bounded [59].

In the DP definition, e^{ϵ} can be seen as an approximation to $1 + \epsilon$ [60] such that an acceptable privacy loss must keep these two values close to each other: $e^{\epsilon} \approx 1 + \epsilon$. However, the multiplicative term e^{ϵ} is preferred, rather than the additive term $1+\epsilon$. Despite the fact that the behavior of e^{ϵ} under multiplication is much more useful ($e^{\epsilon_1}e^{\epsilon_2} = e^{\epsilon_1+\epsilon_2}$) [135], the main reason is that the additive definition allows the existence of a algorithm $\mathcal{M}$ that picks a random user's record from the input database and outputs it, while satisfying $\mathtt{P}(\mathcal{M}(\mathbb{X}) \in \mathbb{Y}) \leq \mathtt{P}(\mathcal{M}(\mathbb{X}') \in \mathbb{Y}) + \frac{1}{\mathtt{size}(\mathbb{X})}$. The multiplicative definition in (2.2.1) rules out the possibility of such algorithm [60]. For the same reason, a DP algorithm with $\delta \geq \frac{1}{\mathtt{size}(\mathbb{X})}$ is not an acceptable algorithm.

When the goal is to approximate a deterministic function $\mathcal{F}$ (*e.g.* the mean or histogram of the dataset), a typical example of $\mathcal{M}$ in Definition 2.1 is the noise addition algorithm, such as a zero-mean Laplace [136] or Gaussian noise [59] with the variance proportionate to the *sensitivity* of $\mathcal{F}$. In general, to be able to calculate the $S_{\mathcal{F}}$, one always needs to know, or at least to bound, the $\mathit{Range}(\mathcal{F})$; otherwise, a DP algorithm cannot be designed.

For example, in a Laplace DP algorithm, the sensitivity of a function is defined as the maximum of the absolute distance $S_{\mathcal{F}} = \max_{\mathbb{X},\mathbb{X}'}|\mathcal{F}(\mathbb{X}) - \mathcal{F}(\mathbb{X}')|$ among all pairs of neighbor datasets $\mathbb{X}$ and $\mathbb{X}'$. Considering a function $\mathcal{F} : \mathbb{R}^{\mathtt{size}(\mathbb{X})} \to \mathbb{R}$, a Laplace algorithm $\mathcal{M}$ that satisfies ϵ-DP is defined as

$$\mathcal{M}(\mathbb{X}) \triangleq \mathcal{F}(\mathbb{X}) + \mathit{Lap}(\frac{S_{\mathcal{F}}}{\epsilon})$$

where $\mathit{Lap}(\frac{S_{\mathcal{F}}}{\epsilon})$ is a random sample drawn from a zero-mean $\frac{S_{\mathcal{F}}}{\epsilon}$-variance Laplace distribution.

DP algorithms give a mathematical bound on the privacy loss for both user-level (local DP) and population-level (central DP) data sharing. A central DP algorithm considers a model where users trust the data aggregator, thus they share their original data with the aggregator. The trusted aggregator then applies transformations (*e.g.* noise addition) to the outcome of a target computation on the collected data before sharing them with other third parties. A local DP algorithm randomly transforms data at the user's side without needing to trust the data aggregator or other third parties [137, 138, 139].

Let $\boldsymbol{x}$ denote the user's original data and $\boldsymbol{x}'$ denote a random value selected from the set of possible values. A local (ϵ, δ)-DP algorithm $\mathcal{M}$ releases $\boldsymbol{y}$ such that $\mathsf{P}(\mathcal{M}(\boldsymbol{x}) = \boldsymbol{y}) \leq e^{\epsilon}\mathsf{P}(\mathcal{M}(\boldsymbol{x}') = \boldsymbol{y}) + \delta$. Thus, in terms of privacy guarantee, local DP ensures that the probability of discovering the true value of the user's data (*i.e.* whether it was $\boldsymbol{x}$ or $\boldsymbol{x}'$) is limited to a mathematically defined upper-bound. However, central DP gives such an upper-bound for the probability of discovering whether a specific user has shared their data or not.

The *central* and *local* are sometimes referred to as *output perturbation* and *input perturbation*, respectively: In the former, the result of a computation is perturbed before releasing it, while in the latter each datum is perturbed, before performing any sort of computations. Although DP algorithms are mostly used for computing aggregated statistics, such as the mean, variance, or histogram of a common attribute among users [140, 141, 142], DP algorithms can also be used for training machine learning models on sensitive datasets while providing plausible deniability guarantees for the participants in the dataset [143, 144, 145]. In [143], authors propose a central DP algorithm, called differentially-private stochastic gradient descent (DP-SGD), that randomly sample a batch of samples from a training dataset and adds random Gaussian noise to the average of the clipped gradients before updating the weights of the model. Similar to DP-SGD, authors in [144] propose a local DP algorithm, called LDP-SGD, where users locally compute the clipped gradients using their private data and then share a randomized version of their gradient which further will be averaged at the server side to update the model.

Differential privacy mainly considers the effect of using or not using a single data point on the result of a computation, and makes no assumption on the distribution of the data. Although DP offers a strong privacy guarantee, adjustable by a well-defined privacy loss parameter ϵ, it only cares about the privacy violation of sharing personal data, and not proposing any explicit notion of the data utility. A cost of having such a strong definition is that DP does not provide a guarantee on the type and amount of *information* that is revealed by sharing the data [71]. Moreover, DP cannot be efficiently utilized in a setting of multivariate data sharing, mainly because of the data correlation and the DP's composition property [136]. Basically, DP algorithms offer how to compute a random noise, $\boldsymbol{n}$, to be added to the original data, $\boldsymbol{x}$, such that an adversary's ability to reconstruct the original data from the received noisy data, $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$, is bounded. As we discussed earlier, for a fixed $\epsilon \geq 0$, the variance of the required noise for DP mainly depends on the sensitivity of the target function. When the function is to release a transformed version of the data, and not a statistic about the data, *e.g.* for a time window of multivariate sensor data, the sensitivity is not only unknown in a dynamic setting but is usually very large.

For example, there will be multiple releases of very similar data patterns in a continual sharing scenario with periodic sensor data, e.g. the gait of a walking person. The composition of T consecutive ϵ-differentially private computation is in order of $(T \cdot \epsilon)$-differentially private [60]. Therefore, to guarantee a desired bound on the privacy loss, the privacy budget ϵ must be decided among the whole period of data release such that, for instance, each data release guarantees a bound of ϵ/T. Thus, when sharing multivariate data, where T is usually a large number, one can only guarantee ∞-differential privacy if we want to keep the utility of data; otherwise, the appropriate noise level would be so high as to be equivalent to complete randomization. Hence, despite the strong privacy guarantee and its usefulness in sharing data bounded to a specific range, DP is not a meaningful notion of privacy for multivariate temporal data sharing.

2.2.2 Crowd-Blending Privacy

Crowd-blending privacy [68] aims to blend the data of every user in a crowd of $l \geq 0$ users in a way that replacing the user's data with any other individual's data in this crowd does not alter the results of the target computation. A necessary condition for satisfying the crowd-blending privacy, for a user's data in a dataset, is the existence of at least $l-1$ other data sample that can be categorized in the same group with the user's data. If the condition is not satisfied for a user's data, then the algorithm must essentially ignore that data and should not use it in the required computation.

Definition 2.2 (**Crowd-Blending Privacy** [68, 146])**.** Given $l \geq 1$, we say a data sharing algorithm $\mathcal{M}$ satisfies l-crowd-blending privacy if for all pairs of neighbor datasets $\mathbb{X}$ and $\mathbb{X}'$ differing in only one sample $\boldsymbol{x}$—such that $\mathbb{X} = \mathbb{X}' \cup \{\boldsymbol{x}\}$—we have

$$\texttt{size}\Bigg(\Big\{\boldsymbol{y} \in \mathcal{M}(\mathbb{X}) : \boldsymbol{y} = \mathcal{M}\left(\{\boldsymbol{x}\}\right)\Big\}\Bigg) \geq l \;\text{ or }\; \mathcal{M}(\mathbb{X}) = \mathcal{M}(\mathbb{X}'). \tag{2.2.2}$$

The Equation 2.2.2 means that (i) for every sample data $\boldsymbol{x}$ in the original dataset $\mathbb{X}$ the corresponding transformed data $\boldsymbol{y}$ is equal to the output of at least $l-1$ other samples, (ii) otherwise $\mathcal{M}$ completely ignores the sample data $\boldsymbol{x}$, hence the ultimate output of the $\mathcal{M}$ does not depend on the presence or absence of $\boldsymbol{x}$.

Overall, every DP algorithm satisfies crowd-blending privacy for every possible $l \geq 1$, hence the crowd-blending privacy can be considered as relaxation of DP [68]. The most important aspect of crowd-blending privacy is that if one performs a *pre-sampling*, before applying the crowd-blending algorithm $\mathcal{M}$ on the collected dataset, the combined pipeline, of first pre-sampling then crowd-blending, provides a DP guarantee [68, 146]. The pre-sampling means to randomly and uniformly choose a fraction of ($p < 1$) percentage of the whole available users. Thus, as the randomness of $\mathcal{M}$ is a necessary condition for a DP algorithm, but not for a crowd-blending algorithm, this pre-sampling introduce the required randomness. This intuitively says that looking at the outcome of $\mathcal{M}$, not only every user's data blends in a crowd of l people (due to crowd-blending), but also the user's data being sampled or not is also not clear (due to pre-sampling).

Notice that the outcome of a crowd-blending algorithm depends not only on the dataset, but also on the desired algorithm $\mathcal{M}$. This makes crowd-blending a more flexible privacy notion than the k-anonymity [66] which is a notion of privacy independent of any specific algorithm, and mainly designed for releasing tabular datasets. Thus, unlike crowd-blending and DP, k-anonymity is not a property of an algorithm, but a property of the dataset. More precisely, a property of a subset of attributes in a tabular dataset, known as *quasi-identifiers*. A tabular dataset satisfies k-anonymity if and only if each combination of quasi-identifiers appears with at least k occurrences [66]. For example if a dataset has three attributes, namely *age*, *gender*, and *disease*, then a 10-anonymity dataset, with *age* and *gender* as the quasi-identifiers, has at least 10 records for each appearing combination of *age* and *gender*. Crowd-blending extends this definition by considering assigning this property to the algorithm that produces the dataset, hence it can also be used in other scenarios than dataset release depending on the required algorithm $\mathcal{M}$.

2.2.3 Inferential Privacy

An IP algorithm minimizes the amount of sensitive information included in the shared data, while keeping a constraint on the amount of distortion that is allowed in the required data.

Definition 2.3 (**Mutual-Information Privacy** [71, 85, 147])**.** Given $\lambda \geq 0$, the distortion measure $\mathcal{D}$, and an observed data $\boldsymbol{x}$, that is informative about two hidden random variable $\mathbf{s}$ (sensitive) and $\mathbf{r}$ (required), we say a data transformation algorithm $\mathcal{M}$ that transforms $\boldsymbol{x}$ to $\boldsymbol{y}$, satisfies a $\lambda-$mutual information privacy if it is a solution of the following optimization problem:

$$\min_{\mathcal{M}:\boldsymbol{x}\to\boldsymbol{y}} \mathtt{I}(\mathbf{s};\boldsymbol{y}) \text{ subject to } \mathtt{I}(\boldsymbol{r},\boldsymbol{y}) \geq \lambda. \tag{2.2.3}$$

where $\mathtt{I}(\cdot,\cdot)$ denotes the mutual information, and the Markov chain $(\mathbf{s},\mathbf{r}) \to \boldsymbol{x} \to \boldsymbol{y}$ is assumed as the underlying structure of data generation process, and the distortion measure can be any deterministic or probabilistic function of the data.

Intuitively, an IP algorithm implies that for a given utility constraint based on λ, among all feasible algorithms, we select the one that minimizes $\mathtt{I}(\mathbf{s};\boldsymbol{y})$. An algorithm with the *perfect utility* tends to release $\boldsymbol{y}=\boldsymbol{x}$ that maximizes $\mathtt{I}(\boldsymbol{r},\boldsymbol{y})$; however this algorithm also maximizes $\mathtt{I}(\boldsymbol{s},\boldsymbol{y})$ that means the *worst privacy*. On the other hand, an algorithm with the *perfect privacy* tends to ignore $\boldsymbol{x}$ and uniformly and randomly release a $\boldsymbol{y}$ that gives $\mathtt{I}(\mathbf{s};\boldsymbol{y}) = \mathtt{I}(\boldsymbol{r},\boldsymbol{y}) = 0$ that is equal to the *worst utility*.

IP algorithms propose establishing a trade-off between the utility of the required information, and the privacy of the sensitive information such that the shared transformed data be as informative as possible about the required information, while revealing as little as possible the sensitive information [147]. Thus, IP algorithms are aimed to reduce the possibility of inferring user's sensitive information from the shared data, while keeping the possibility of inferring non-private required information from data at an acceptable level [85, 148].

Since the optimization problem defined in (2.2.3) is a non-convex [147] problem, as a practical alternative we can rewrite (2.2.3) as a minimax optimization

$$\min_{\mathcal{M}:\boldsymbol{x}\to\boldsymbol{y}} \max_{\mathcal{A}:\boldsymbol{y}\to\boldsymbol{s}} \mathtt{I}(\mathbf{s};\boldsymbol{y}) + \max(0, \lambda - \mathtt{I}(\mathbf{r};\boldsymbol{y})), \tag{2.2.4}$$

where $\mathcal{A}$ is an adversarial algorithm that is optimized to infer the sensitive information. Thus, in turn and repeatedly, we can optimize $\mathcal{M}$ and $\mathcal{A}$ in minimax game. Notice that, when we are optimizing $\mathcal{A}$, $\mathcal{M}$ is fixed and so the second term in (2.2.4) is ignored by $\mathcal{A}$.

In a practical setting, it is not straightforward to target a desired λ for the required utility. Therefore, a similar but more straightforward formulation of (2.2.4) is:

$$\min_{\mathcal{M}:\boldsymbol{x}\to\boldsymbol{y}} \max_{\mathcal{A}:\boldsymbol{y}\to\boldsymbol{s}} \mathtt{I}(\mathbf{s};\boldsymbol{y}) - \alpha\mathtt{I}(\mathbf{r};\boldsymbol{y}), \tag{2.2.5}$$

where $\alpha > 0$ is a hyper-parameter that can be used to establish a desired privacy-utility trade-off. In Chapter 4, we provide some examples of such trade-offs where we regularize the loss function of a DNN based on such hyper-parameter during training.

Similar to DP, a data aggregator can be trusted or not, hence data transformation can be done at the user's side or on the collected data from a population [149, 88, 86]. Unlike DP, IP algorithms do not guarantee an exact statistical upper bound for the privacy loss incurred by each data release. The main reason is that the statistical bounds in DP algorithms, ϵ and δ, depend on the chosen noise and the sensitivity of the desired function. However, the privacy metrics used in IP algorithms are mainly based on the theory of *Shannon information content* [150] where for the exact computation of the provided upper-bounds, we need to know the exact probability of each possible outcome to compute mutual information; a requirement which is almost impossible to satisfy for real-world data. Hence, IP algorithms usually provide an approximation of the provided privacy guarantees [62, 89]. As, in practice, the data distribution is estimated from an available dataset, the effect of the discrepancy between the estimated empirical distribution and the true data distributions can be a great risk to privacy, which should be taken into account when offering IP algorithms [151].

In practice, we need an empirical estimation of mutual information to find an approximate solution to the problem in Equation 2.2.5. Poole *et al.* [152] provide a summary and evaluation of several recent methods of mutual information estimation. Adversarial estimation using a parameterized model is one of these common approaches that is used in the literature [86, 88, 153]. As $\mathtt{I}(\boldsymbol{s} \mid \boldsymbol{y}) = \mathtt{H}(\boldsymbol{s}) - \mathtt{H}(\boldsymbol{s} \mid \boldsymbol{y})$ and in a fixed dataset $\mathtt{H}(\boldsymbol{s})$ is fixed (*i.e.* does not depend on the algorithm $\mathcal{M}$). Thus the estimation of $\mathtt{I}(\boldsymbol{s} \mid \boldsymbol{y})$ is reduced to the estimation of $\mathtt{H}(\boldsymbol{s} \mid \boldsymbol{y})$ which can be achieved via a parameterized model, *e.g.* a neural network, that takes $\boldsymbol{y}$ as input and produces $\hat{\boldsymbol{s}}$ as an estimation to the true $\boldsymbol{s}$ trained via log-loss [147]. In Chapter 4 we discuss more on such adversarial estimation and use it for training an autoencoder that aims to transform data such that it (approximately) minimizes sensitive information.

2.3 Privacy-Preserving Data Sharing

In Table 2.1 we provide a summary of algorithms that have been used for privacy-preserving data sharing. In terms of the privacy concern, a privacy-preserving algorithm aims to release data such that the released data does not include sufficient information either (i) to confidently *reconstruct* the original data, or (ii) to discover a *sensitive latent information* from the released data, or (iii) to realize whether a specific user's data is part of the released data or not (*i.e. membership* inference). Regarding the utility of the released data, a privacy-preserving algorithm should preserve information with respect to a required task such as calculating the *sum*, estimating a *histogram*, or discovering a *required latent information.* Privacy-preserving algorithms are either run at the

Table 2.1: A summary of privacy-preserving algorithms for data sharing. U and S refer to the user and server side, CDP and LDP refer to the central and local differential privacy, and IP refers to the inferential privacy.

Work	*Algorithm*	*Privacy*	*Utility*	*Side*	*Data Type*	*Bound*
[22]	perturbation	membership	sum	S	binary	CDP
[154]			histogram	S	categorical	CDP
[137, 141, 155, 156]			histogram	U	categorical	LDP
[92, 157, 158]		reconstruction	latent info.	U	real-valued	LDP
[26]						–
[25]	synthesis	latent info.	latent info.	S	real-valued	–
[62, 159]		membership	latent info.			CDP
[80]	filtering	latent info.	latent info.	U	real-valued	–
[84]				S		–
[82]				U		IP
[88, 86, 160]	sanitization	latent info.	latent info.	S	real-valued	IP
[161]				U		IP
[162]				U		–
[149, 163, 73, 74]				S		–

User side or the *Server* side, but not all of them are based on the well-defined, DP or IP, privacy bounds.

Achieving the perfect preservation of both privacy and utility in multivariate data sharing is a challenging task. Therefore, there have been many efforts to design and evaluate algorithms that allow us establishing a trade-off between privacy and utility [13, 19, 164, 85]. In the following, we discuss and compare the related privacy and utility preserving algorithms, grouped into four categories: *perturbation*, *synthesis*, *filtering*, and *sanitization*.

2.3.1 Perturbation

Perturbation generally means adding a carefully computed noise to the original data. Apple proposes a local DP algorithm to collect personal data for getting insights into the most popular items among their users [155]; *e.g.*, the words or emojis that are trending or the health categories that are most popular among users. Each user's device, once per day, chooses the latest temporal data point (*e.g.* the latest typed emoji), applies local randomization to it [22], and sends it over an encrypted channel to the server. A mediator server is used to remove the communication identifiers and any other data that can associate a data point to a specific user. Such a mediator can also buffer the users' shared data and later shuffle them before sending them to the untrusted server. The buffering and shuffling operations can protect the system against timing attacks and also amplify the guaranteed DP privacy bound [139, 158].

An important concern regarding the use of local DP algorithms is about privacy leakage that can occur if data is collected repeatedly. Although the privacy guarantee for a single data point may seem appropriate, for instance, if the $\epsilon \leq 1$, the overall privacy loss per device is increasing over time if the user's data points do not significantly change or if they are correlated. For example, it is a reasonable assumption to say a user may send similar emoji multiple times. Depending on the value of ϵ, after a number of rounds, the server has enough noisy versions of the same data to confidently reconstruct the original data. The privacy loss upper-bound proposed by local DP follows the composition theorem [60] which states that the privacy guarantee gets weaker by a factor of the number of iterations.

A technique called *memoization* can be used to remedy the effect of repeated data collection [137, 141]. In memoization, a user's data point is randomized once at the user's side, then at each iteration, the user sends a randomized instance of that fixed value to the server. Randomization is done by dividing the promised privacy budget ϵ among T iterations, and by adding the Laplace noise with a variance of $\frac{\mathrm{T}}{\epsilon}$. Google [137] and Microsoft [141] use a local DP algorithm with Memoization for the estimation of mean or histogram of the user's daily usage statistics; across millions of their users. Memoization has some limited applications such as calculating the mean or the histogram, and it's meaningful if the user's data remains approximately the same, or varies around a small number of values. Fan *et al.* [154] propose to exploit the domain knowledge: after adding Laplacian noise, the authors perform a post-processing stage to release more utility-preserving data for sharing location data in multiple iterations. Joseph *et al.* [156] propose to repeatedly recompute a statistic with an error that decays proportional to the number of times the data changes significantly rather than the total iterations.

In sharing multivariate data, DP cannot offer a practical algorithm. Hence, privacy-preserving algorithms for time series sharing consider privacy models based on domain-specific privacy and utility definitions. To hide sensitive patterns in time series a crafted, privacy-preserving algorithms add a noise that is either independent or correlated [165, 162], to each time window of the time series. As independent and identically distributed noise can be easily removed from correlated time series [26] to reduce the risk of information leakage, the correlation between noise and original time series should be indistinguishable [83, 166]. Low variance additive random noises can be filtered out using a method called random-matrix filtering, while noises with high variance completely destroy the utility of data [167].

2.3.2 Synthesis

When the primary goal of collecting personal data is to train or evaluate machine learning models, one can build a synthetic data generation model based on the distribution of the original dataset. A *synthetic* dataset maintains some required information of the original data, but not all the information that can be used to re-identify users or are related to the user's privacy. For example, to protect health records, synthetic medical datasets can be published instead of the real ones using generative models trained on sensitive real-world medical datasets [62]. Zhang *et al.* [168] propose an *obfuscation* method that generates and releases synthetic data (*e.g.* HTTP requests) alongside real data to ensure that the service provider cannot distinguish which incoming data are real and which are generated.

Optimization methods can be used to generate and publish an entirely new time series from the original ones by considering some predefined constraints on the type of information that can be inferred [25]. Generative adversarial networks [114] can be used to approximate the data distribution to generate new synthetic data that is similar to the existing one [169]. To provide a privacy guarantee, generators can be trained under the constraint of DP [159, 170] or with constraints on the type of information that should be unsynthesized in the data [25]. However, these generative algorithms are mostly usable for dataset publishing by a data aggregator [62], not for online data transformation at the user side.

2.3.3 Filtering

In some applications of sensor data analysis, users only want to filter some sensitive time windows of their data instead of perturbing the whole time series [81]. Thus, an appropriate algorithm should apply an amount of perturbation to each time window of a time series proportionate to the structure of that time window. *Filtering* algorithms can be used to remove time windows that include sensitive information, while releasing time windows that are required and not sensitive without any modification. For example, users of a smart home must give the aggregate electrical usage to the service provider, but might want to keep the information related to their eating patterns private (*e.g.* microwave usage).

Erdogdu *et al.* [82] propose to estimate the correlations between time series to be released and a private variable using probabilistic graphical models to filter out the time windows that are highly correlated to that private data. An algorithm called MaskIt [80] releases location time series when users are at a regular workplace and suppresses location data when users are in a sensitive place, such as a hospital, by building a Markov chain on a pre-defined set of sensitive locations. Saleheen *et al.* [84] suggest a dynamic bayesian network model to replace sensitive time windows that indicate users' stress, while keeping non-sensitive time windows corresponding to their walking periods. Overall, when we are facing with multivariate time series of fine granularity, algorithms based on the probabilistic model are not tractable, as it is not easy to find an accurate model of correlation between the original data and the noise needed for the desired utility-privacy trade-off. Hence, hiding sensitive patterns without excessively perturbing the non-sensitive ones is very challenging when general time series perturbation approaches are extended to sensor data.

2.3.4 Sanitization

Sanitization is the process of removing *sensitive information* before publishing data which is formulated through a minimax game[88, 161] between two players: a user who wishes to hide their sensitive data and a malicious app that seeks to find the sanitized sensitive data. For example, by smoothing the signal and reducing the sampling rate of time series data, users can protect such sensitive inferences that can be made on fine-grained data, but not from the coarse-grained version of the data [162].

The capacity of deep neural networks in extracting useful features helps to capture the main factors of variation in the data and to identify and obscure sensitive patterns in the latent representation [86], as well as during the reconstruction from the extracted low-dimensional representation [161, 171]. Thus, using deep neural networks, the original data can be transformed into a lower dimensional [86, 149, 160] or the same dimensional [88, 161] data such that the amount of sensitive information in the data is reduced while non-sensitive information is preserved for utility.

2.4 Learning from Distributed Data

As we discussed in Section 2.2, DP algorithms are not practical to be used for releasing multivariate data, and IP algorithms require having knowledge of the true data distribution to provide an exact statistical guarantee. Particularly, such privacy-preserving data sharing algorithms might not be useful in situations where data is highly sensitive and informative about the user's identity (*e.g.* web-browsing data). To tackle this problem, an alternative approach is that data does not need always to leave the user's device if we only need to learn some specific information from that data. Federated learning is built upon this approach of doing on-device training through the idea of "bringing the code to the data, instead of the data to the code" [91]. Thus, instead of sharing the raw data, or sharing a randomized or transformed version of the data, users train the received machine learning model on their private data and share the updated parameters of that model. In federated learning, DP algorithms can be used to add noise to the model's parameters, instead of directly adding noise to the data, which is shown to be a more utility preserving solution [144]. To ensure that the updated model remains encrypted even in the server's memory, one can use the secure aggregation through multi-party computation [91, 90]. Thus, instead of performing a data transformation and releasing the transformed data for further computations, users can keep their data and choose whether they want to participate in a specific computation or not.

The Draw and Discard algorithm [92] is a combination of federated learning and differential privacy that encompasses decentralized model training for generalized linear models. The Draw and Discard consists of two components: On the server's side, it sends a randomly chosen model instance (among k available models) to the user that is trained locally at the user's side. Upon receiving an updated model instance from a user, the server randomly replaces it with one of the k existing instances. On the user side, users update the linear model on their data, and add Laplacian noise to the model parameters to satisfy differential privacy. Draw and Discard assumes that the features of the model are all uncorrelated. As a consequence, for achieving model-level privacy, where most features are correlated to each other, substantially more noise needs to be added to the updated model before sending the model to the server.

A collaborative system for training deep neural networks in mobile environments is proposed by Liu *et al.* [172] that enables multiple agents to train a model only by sharing partial parameters with each other; arguing that privacy is protected by keeping data locally and only share a small fraction of the parameters to the server at each round. However, even in such cases, if a system does not satisfy the DP requirements, then there is the risk of model inversion attacks that reconstruct data based on the shared parameters [173]. To protect the privacy of the patients, a federated learning system for brain tumor segmentation is proposed in [157] in a setting where each hospital owns a dataset of MRI scans. Shariff *et al.* [174] discuss that adhering to the standard notion of DP implies high data perturbation. Thus, they use a relaxed notion of DP, called "joint differential privacy", that allows using the non-privatized data at time t , while guaranteeing that all interactions with all other users at time points $t' > t$ have very limited dependence on that user data at time t.

2.5 HAR Datasets

In Table 2.2, we summarize the details of motion-sensor datasets, for human activity recognition, that are used in this thesis.

Opportunity [97] is composed of the collected data of 4 users and there are 18 gestures classes. Each record in this dataset comprises 113 sensory readings from various types of body-worn sensors like accelerometer, gyroscope, and magnetometer.

Skoda [96] is collected by an assembly-line worker in a car production company wearing 19 accelerometer sensors on his right and left arm and performing a set of pre-specified experiments.

Hand-Gesture [98] includes data from accelerometer and gyroscope sensors attached to the upper and lower arm. There are two users performing 12 classes of hand movements. Each record in this dataset has 15 real-valued sensor readings.

Utwente [99] includes the data of 10 participants performing several activities, including potentially sensitive smoking activity, while wearing a smartphone on their wrist. Accelerometer, gyroscope, and magnetometer data are collected. The whole dataset is publicly available in a single file with activity labels only. We divide the dataset into 80% training and 20% validation.

UCI-HAR [100] is a widely used dataset [101, 102, 103, 117] of 30 users performing 6 activities. Accelerometer and gyroscope data were collected by a smartphone worn on the waist. Data from 21 users are used for training and that of the other 9 users for testing purposes.

MobiAct [175] contains the accelerometer, gyroscope, and orientation data collected from users who kept a smartphone in their front pocket and performed different types of activities. We use the data of 55 users, including 41 males and 14 females, who have performed falls alongside other normal activities.

Our dataset, **MotionSense**, is collected with a smart-phone kept in the users' front pocket. A total of 24 users (14 males and 10 females) performed 6 activities in 15 trials in the same environment and conditions at the campus of Queen Mary University of London. It includes accelerometer and gyroscope data. Among these HAR datasets, only MobiAct and our MotionSense include users' attributes such as gender, age, weight, and height which are highly useful for evaluating privacy-preserving algorithms that need multi-label datasets. MobiAct dataset has been released, while we were collecting MotionSense dataset, but we were not aware of that. Nevertheless, because of sharing some similarities such as the type and on-body location of the device, MobiAct and MotionSense can be combined for evaluating HAR models, as in Appendix A we show an example of such evaluation.

For each user, the data collection had been commenced by collecting the user's attributes: gender, age, weight, and height. Then, we provided the user with a specific smartphone (iPhone 6) and asked them to keep the phone in their trousers' front pocket during the experiment. All the users were asked to wear flat shoes and tight trousers. The shoes and trousers were not chosen by us, but their own clothes. We asked each user to perform 6 different activities (walking, jogging, sitting, standing, stairs-up, and stairs-down) around the Mile End campus of Queen Mary University of London.

In each trial, we set up the phone and gave it to the user, then we stood in a corner. The user pressed the start button of the Crowdsense app [176], and put it in their trousers' front pocket, and performed one of the defined activities. We asked them to do it as naturally as possible, like their everyday life. At the end of each trial, they took the phone out of their pocket and pressed the stop button. MotionSense is published at `https://github.com/mmalekzadeh/motion-sense`.

Table 2.2: Details of the datasets for human activity recognition based on motion sensors (A: accelerometer, G: gyroscope, and M: magnetometer) that we use throughout this thesis. The *null* class refers to data that cannot be mapped to a known behavior. Only two datasets include user's attributes: gender, age, weight, and height.

#	**Opportunity** [97]	**Skoda** [96]	**HandGesture** [98]	**Utwente** [99]	**UCI-HAR** [100]	**MobiAct** [177]	**MotionSense** (**Ours**)
0	null	null	null	—	—	—	—
1	open door1	write notes	open window	walking	walking	walking	walking
2	open door2	open hood	close window	stairs-down	stairs-down	stairs-down	stair-down
3	close door1	close hood	water a plant	stairs-up	stairs-up	stairs-up	stairs-up
4	close door2	check front door	turn book	standing	standing	standing	standing
5	open fridge	open left f door	drink a bottle	sitting	sitting	sitting	sitting
6	close fridge	close left f door	cut w/ knife	jogging	lying	jogging	jogging
7	open washer	close left doors	chop w/ knife	cycling	—	jumping	—
8	close washer	check trunk	stir in a bowl	typing	—	falling	—
9	open drawer1	open/close trunk	forehand	writing	—	—	—
10	close drawer1	check wheels	backhand	eating	—	—	—
11	open drawer2	—	smash	smoking	—	—	—
12	close drawer2	—	—	—	—	—	—
13	open drawer3	—	—	—	—	—	—
14	close drawer3	—	—	—	—	—	—
15	clean table	—	—	—	—	—	—
16	drink cup	—	—	—	—	—	—
17	toggle switch	—	—	—	—	—	—
Users	4	1	2	10	30	61	24
Features	113	57	15	9	6	9	6
Sampling Rate (Hz)	30	30	30	50	50	50	50
Sensors	A, G, and M	A	A and G	A, G, and M	A and G	A,G, and M	A and G
User' Attribute	No	No	No	No	No	Yes	Yes

2.6 Summary

We have discussed the recent progress in the field of data privacy with a focus on the algorithms proposed for time series data sharing. While existing works are proved useful for many applications, there still are some challenges in applying them to behavioral data generated by users of mobile and wearable devices.

First, most of the previous works are only useful in an offline setting where we first collect all the data at one place and then perform transformations (*e.g.* sanitization or anonymization), however in many real-world situations we need algorithms that can be shipped to the user's side for performing online data transformations. For example, it might not be a realistic assumption to design a trusted mediator system that collects sensor data produced by all users of an activity recognition app every day, and then performs anonymization before sharing users' data with the app for receiving some health monitoring statistics. Similarly, it is hardly imaginable to have a trusted mediator for collecting and anonymizing all users' web browsing histories sharing them with a recommendation app to propose some personalized contents.

Second, previous works are mostly focused on the extraction of a population-level aggregated statistic from personal data, but for many situations we need to extract user-level information and propose personalized services based on their behavioral data. In some personalized settings, *e.g.* in continually sharing multivariate sensor data, central DP algorithms are not applicable and local DP algorithms cannot achieve any meaningful utility. Thus, we need to think of a more practical notion of privacy. For example using inferential privacy with empirical guarantees, where privacy loss is quantified based on the empirical estimation of information leakage. Although the guaranteed privacy by such empirical algorithms is weaker than a pure IP algorithm or a local DP algorithm, these empirical algorithms can provide some acceptable privacy protection that might be useful in situations that there are no other choices than having no privacy at all.

Therefore, in this thesis, we focus on machine learning algorithms that can be run at the *user side* and can be generalized to unseen users who did not contribute training data. The aim is to cover the shortcoming of centralized approaches that first need to collect all users' data, then release a transformed version of the collected dataset to the untrusted parties. In the proposed work in this thesis, we do not assume the existence of a trusted data aggregator to perform anonymization for end users. We only assume having access to a public dataset for training a generalized algorithm. It is much more reasonable to design a privacy-preserving machine learning algorithm that can be trained on a dataset collected from hundreds of users, and then be used by thousands of other users to perform anonymization locally.

In addition, we argue that when it comes to multivariate data with characteristics we need algorithms that are adaptive and robust to changes so they can be used across different users and devices. While making an algorithm adaptive to changes in the dimension of their input data does not make it a privacy-preserving algorithm per se, it can enable the design and deployment of more flexible privacy-preserving algorithms. For instance, if we want to allow users to have more control over the data they are sharing with an activity recognition app, an adaptive algorithm allows apps to continue their service even if the user can choose the type of sensors and sampling rate of the data.

Chapter 3

Sensitive Data Replacement

3.1 Overview

A typical setting in health and wellness monitoring via wearable devices is an *app* that monitors the *user*'s activities through time, reporting useful statistics to them alongside some fitness or health advice [**?** 8, 99, 34]. Basically, not all the user's activities are *required* to be recognized by the app to achieve the desired utility, and more importantly, some of the non-required activities might be *sensitive* to the user's privacy. Ideal privacy and utility preserving algorithm should allow the app to accurately recognize the required activities while preventing it from the recognition of sensitive activities [178, 84]. To achieve such a desire, we propose to utilize the capacity of activities that are neither required nor sensitive; in the following, we will refer to these as *neutral* activities.

For example, Figure 3.1 shows 11 samples of accelerometer time-series from a smartwatch worn on the user's right wrist [99]. Considering an app that counts a user's daily steps, the user may want this app to only be able to exclusively infer activities that are required for step counting task (*e.g.* walking, jogging, and stair-stepping), and not inferring activities such as smoking or typing that may be considered sensitive. The idea is to automatically detect and replace data patterns that reveal sensitive activities with the same dimensional data that simulates neutral activities that do not affect the step counter's utility (*e.g.* standing or sitting that are not related to the step counting). In this chapter, we propose Replacement AutoEncoder (RAE) to transform data patterns that correspond to sensitive activities into some patterns that are more observed in neutral activities, while keeping the patterns corresponding to required activities as unmodified as possible[1].

[1]Code and data to reproduce results are publicly available at `https://github.com/mmalekzadeh/`

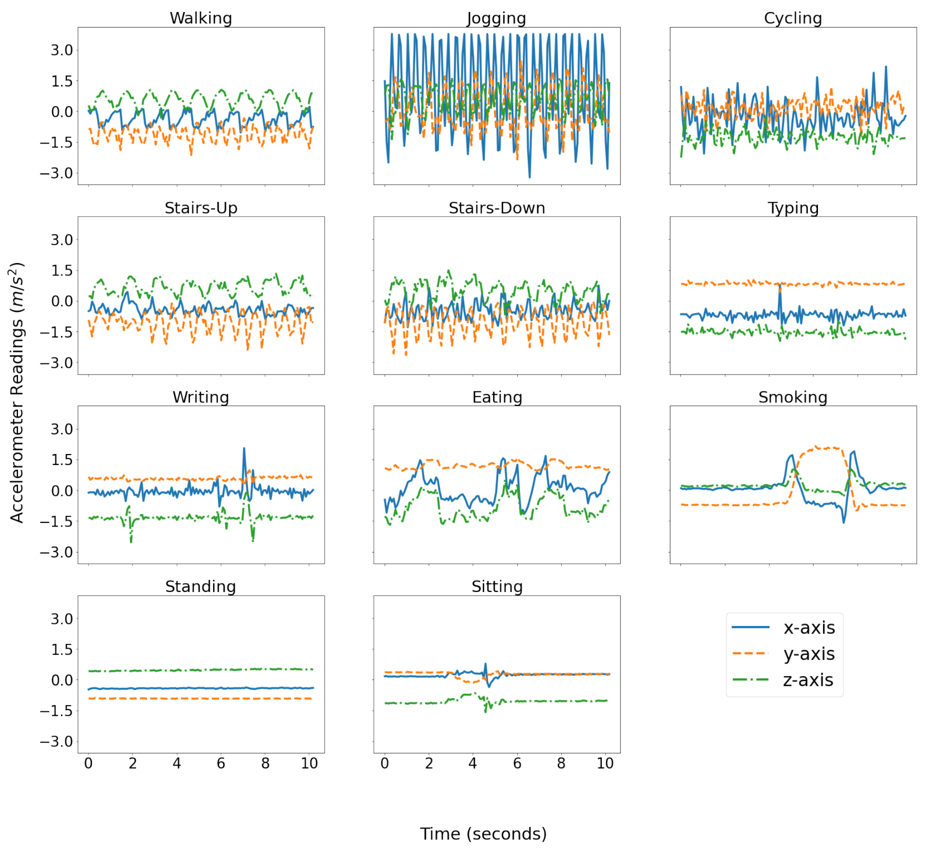


Figure 3.1: Three-axis (x, y, z) accelerometer data of a smartwatch worn on the right wrist, from UTwnete dataset [99].

3.2 Autoencoders

The performance of a classification task depends on the data type, the data representation, and the quality of the extracted features on which a classifier is applied. When using classical machine learning algorithms for classification, such as decision trees or nearest neighbors, we need to first devise an appropriate feature extractor to pre-process multi-dimensional raw sensor data, and then feed them into the classifier. On the other hand, a *deep neural network* (DNN) can automatically learn representations of data that capture relevant features for the classification tasks. The key aspect of representation learning is the extraction of features from data and discover the informative representation needed for the desired task.

DNNs are representation learning architectures with multiple levels of representation, developed by composing a multi-layer stack of non-linear modules

`replacement-autoencoder`

that each transform the representation at one level into a representation at a higher, slightly more abstract level [179]. The hidden layers of a deep neural network (starting with the raw input) can potentially learn more abstract features at higher layers of representations and reuse a subset of these features for each particular task. The earlier layers of a deep neural network contain more generic features that should be useful for many tasks, but later layers become progressively more specific to the details of the classes relevant to the desired task [180, 181]. Thus, feeding such representations to a classifier helps to achieve better generalization [181]. When compared to other machine learning approaches, DNNs have shown superior performance in processing data streams generated by mobile and wearable sensors [116, 102, 101, 182, 103, 119, 183].

An autoencoder is a deep neural network that composes of an *encoder* $\mathcal{F}^e(\cdot;\theta^e)$ and a *decoder* $\mathcal{F}^e(\cdot;\theta^e)$, respectively parameterized by the parameter sets θ^e and θ^d. As an unsupervised feature extractor, an autoencoder is trained to first *encode* the input $\boldsymbol{X}$ into a latent representation $\boldsymbol{z} = \mathcal{F}^e(\boldsymbol{X};\theta^e)$, and then to output $\boldsymbol{Y} = \mathcal{F}^d(\boldsymbol{z};\theta^d)$ as the reconstruction of the input by *decoding* that latent representation. The parameters of the autoencoder are optimized to minimize average reconstruction error $\mathcal{L}(\boldsymbol{X},\boldsymbol{Y})$:

$$\theta^e, \theta^d = \underset{\theta^e,\theta^d}{\arg\min} \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{X}^i, \boldsymbol{Y}^i)$$

where $\boldsymbol{X}^i$ is the i-th sample in the dataset, $\boldsymbol{Y}^i$ is the produced output for $\boldsymbol{X}^i$, and $\mathcal{L}$ is a loss function (*e.g.* mean squared error) that measures the discrepancy between original data and its reconstruction, over all provided training examples [184].

By constraining the latent feature vector $\boldsymbol{z}$ to have a lower dimension than input data $\boldsymbol{X}$, an autoencoder is forced to learn and capture important features of the underlying data-generating distribution. This bottleneck 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 [185, 180]. For example, if an autoencoder has just one hidden layer with a linear activation function and the loss function is the mean squared error, then the k hidden units of that hidden layer learn to project the input in the span of the first k principal components of the data [186]. Using non-linear activation functions, one can extend the learned feature set beyond just principal components, and even capture more complex and non-linear features [187].

Another effective way to train an autoencoder is to randomly add noise to the original input and force the model to refine it in the reconstruction, called *denoising* autoencoder [188]. In this way, a well-trained autoencoder captures prominent and desired patterns in the data and ignores noise or undesired patterns [188].

3.3 RAE Methodology

Let us elaborate the assumptions about the three categories of *patterns* in sensor time series that can be mapped into corresponding human activity.

- *Required*: A set of patterns that help the app to recognize activities which the user gains utility from sharing with the app.
- *Sensitive*: A set of patterns that help the app to recognize activities that users wish to keep private and should not be revealed to the app. These activities are sufficiently sensitive that users would wish to prevent inference that they have undertaken any activities within this set even if the app cannot know which specific one.
- *Neutral*: A set of patterns that correspond to activities that are neither sensitive to the user's privacy nor required for gaining utility from the app. This set may include patterns that cannot be mapped to a known or meaningful activity.

Our assumptions are that (i) we know what activities are *required* for the app to provide the promised utility, (ii) we know what activities are *sensitive* to the users, and (iii) every activity that is not listed as *required* or *sensitive*, will be considered as a *neutral* activity. We assume B classes of activities, $\boldsymbol{a} = \{a_1, ..., a_i, ..., a_j, ..., a_B\}$, are divided into three categories: (i) *Required*, $\boldsymbol{r} = \{a_1, ..., a_i\}$, (ii) *Sensitive*, $\boldsymbol{s} = \{a_{i+1}, ..., a_j\}$, and (iii) *Neutral*, $\boldsymbol{n} = \{a_{j+1}, ..., a_B\}$. The focus is on the classification of the user's activities through pattern recognition in sensor time series, especially using deep neural networks [102, 101, 116].

Let $\boldsymbol{X} \in \mathbb{R}^{W \times H}$ denote a time window of length W that includes H sensor streams (*i.e.* variables). Let $\boldsymbol{X}$ contain patterns that can be classified into a known activity class. Let the training dataset include labeled sample time windows, each belonging to one of the following categories: *required*, *sensitive*, or *neutral*. Let $\mathbb{X} = \{\mathbb{X}^{required}, \mathbb{X}^{sensitive}, \mathbb{X}^{neutral}\}$ be the *input* dataset and $\mathbb{Y} = \{\mathbb{X}^{required}, \mathbb{X}^{sensitive \rightarrow neutral}, \mathbb{X}^{neutral}\}$ be the *output* dataset, with a one-to-one relationship between each $\boldsymbol{X} \in \mathbb{X}$ and $\boldsymbol{Y} \in \mathbb{Y}$ explained in Figure 3.2. Basically, the only difference between $\mathbb{X}$ and $\mathbb{Y}$ is that data samples of *sensitive* classes, $\mathbb{X}^{sensitive}$, are randomly and uniformly replaced with data samples from one of the *neutral* classes, $\mathbb{X}^{neutral}$, to build $\mathbb{Y}$. Therefore, $\mathbb{Y}$ contains only samples

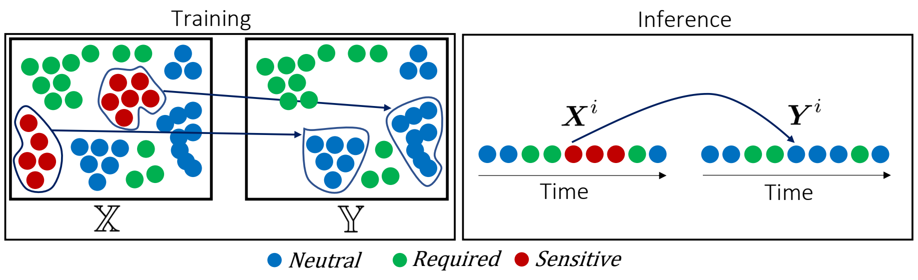


Figure 3.2: Circles represent time windows in the input ($\mathbb{X}$) and output ($\mathbb{Y}$) datasets for training the RAE. We first make a copy of the original input dataset and replace every *sensitive* time window with a randomly chosen *neutral* one to prepare the output dataset for training the RAE. Then, the RAE is trained to transform each $\boldsymbol{X}^i \in \mathbb{X}$ to the corresponding $\boldsymbol{Y}^i \in \mathbb{Y}$. At inference time, RAE can replace unseen *sensitive* time windows with data that simulates *neutral* ones.

from the *required* and *neutral* classes. The RAE is then trained to transform each $\boldsymbol{X}$ to the corresponding $\boldsymbol{Y}$, subject to a loss function, $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y})$, which calculates the difference between the input of the RAE and its corresponding output.

We define the replacement algorithm, $\mathcal{M}$, a *privacy-preserving algorithm* if its outcome limits the possibility of recognizing the user's *sensitive* activities using a machine learning classifier trained by the adversary. Thus, if $\boldsymbol{Z} \in \mathbb{R}^{W \times H}$ is the best possible replacement for a sensitive time window $\boldsymbol{X}$, then, $\mathcal{M}$ aims to implement the following operation:

$$\boldsymbol{Y} = \mathcal{M}(\boldsymbol{X}) = \begin{cases} \boldsymbol{Z} & \text{if } \boldsymbol{X} \text{ contain sensitive data patterns,} \\ \boldsymbol{X} & \text{otherwise,} \end{cases} \tag{3.3.1}$$

Let $\mathcal{M}(\cdot;\theta)$ be a practical implementation of $\mathcal{M}$: an autoencoder with parameter set θ. Let $\mathcal{L}(\cdot,\cdot)$ be the autoencoder's loss function. Then, the optimal parameter set for the $\mathcal{M}(\cdot;\theta)$ is defined as

$$\theta^* = \arg\min_{\theta} \mathcal{L}\Big(\mathcal{M}(\mathbb{X};\theta), \mathbb{Y}\Big), \tag{3.3.2}$$

which is achieved through a neural network optimization process.

The idea of RAE is inspired from the *denoising autoencoder* [184] that takes a noisy data sample, $\boldsymbol{X}^{noisy} = \boldsymbol{X} + \boldsymbol{N}$, and removes the noise, $\boldsymbol{N}$, by producing a reconstruction $\boldsymbol{Y} \approx \boldsymbol{X}$. To train a denoising autoencoder, one only needs to make a pair of datasets $\mathbb{X}$ and $\mathbb{Y}$, the original dataset and the noisy version of it, by using a desired noise addition function. It is also based on the information bottleneck method [189] that considers a setting where we want to compress $\boldsymbol{X}$ into another smaller-sized representation, while maintaining as much information as possible that it carries about a hidden correlated factor. The information bottleneck in the hidden layers of the autoencoder forces the RAE to put more attention on the descriptive and useful data patterns in order to better reconstruct data and generalize to unseen data [181].

Thus, as a summary of the promised utility and piracy to the user, RAE transforms each time window of sensor data, with a pre-defined length (*e.g.* 2 seconds), such that:

- *Utility.* First, the implemented classifier for activity recognition should correctly recognize time windows that correspond to the user's required activities with an accuracy similar to what we expected to get if we could have released the original data (*i.e.* high true positive rate for required activities). Second, time windows of non-required activities should not be transformed such that the activity classifier mistakenly recognizes them as one of the required activities (*i.e.* very low false-positive rate for required activities).

- *Privacy.* First, sensitive time windows should be transformed such that the classifier recognizes them as one of the neutral activities (*i.e.* high false-negative rate for sensitive activities). Second, required time windows should not be recognized as one of the sensitive activities by the classifier (*i.e.* very low false-positive rate for sensitive activities).

- *Adversary.* First, we assume each *time window* only contains patterns that correspond to one activity. Second, we assume that each time window is sampled i.i.d. and thus we do not address situations where there might be any temporal or long-term correlation between different time windows. These two are mainly due to the limitation that we face in the publicly available HAR datasets (also mentioned in Section 2.1.3). Third, we assume that adversaries only have access to the data released by the RAE to launch an attack, and do not consider adversaries that can take advantage of other side information that they can collect about users' behavior. Fourth, we assume that adversaries are computationally bounded and use empirical machine learning models to perform the classification of the user's data.

Our assumptions are similar to those considered in [84] where instead of releasing the original sensitive time window, a Bayesian model is used to find the "most-plausible" non-sensitive time window to be released. However, the algorithm in [84] ignores neutral activities and it is an offline algorithm that assumes the availability of all time windows before starting the replacement procedure. For this reason, they maintain a mapping database to be used in the replacement phase, that stores sensor time windows of different lengths for each possible state. Our proposed RAE can be used in an online transformation setting, it does not need storing an auxiliary database, and by using neutral activities for replacement it does not damage the utility of required activities.

3.4 Evaluation

To evaluate the RAE, we use four benchmark datasets of activity recognition including at least 10 different labels; called Opportunity [97], Skoda [96], Hand-Gesture [98], and Utwente [99] (see Table 2.2). For Opportunity, we use four trials as the training data, and consider the last trial as the testing data. For other datasets, we consider 80% of the data as the training set and the rest as the testing set.

3.4.1 RAE Performance

We implemented RAE with the following settings. We use seven fully-connected layers with the number of neurons equal to N, $\frac{N}{2}$, $\frac{N}{8}$, $\frac{N}{16}$, $\frac{N}{8}$, $\frac{N}{2}$, and N, respectively where $N = W \cdot H$ (except for the Hand-Gesture dataset with a lower dimensionality where the three middle layers are $\frac{N}{3}$, $\frac{N}{4}$, $\frac{N}{3}$). For all datasets, we consider a 1-second time window, $W = 30$. All the experiments are trained on 30 epochs with batch size 128. The activation function for the output layer is *linear* and for the input and all of the hidden layers is Scaled Exponential Linear Unit [190]. In our experiments, to retain the overall structure of the reconstructed data, we set $\mathcal{L}$ in Eq. (3.3.2) as the point-wise mean square error function.

To evaluate the privacy loss and utility of the RAE's outcomes, both the raw sensor data and the transformed data inputs to a state-of-the-art deep neural network classifier, as the assumed app, and F1-score are calculated in Table 3.1 and Table 3.2. We use F1-score as the evaluation metric because it takes both false positives and false negatives into account and is a better metric when there are imbalanced classes as in the dataset.

The results show that the utility is preserved for non-sensitive, $\boldsymbol{r}$ and $\boldsymbol{n}$, classes while recognizing sensitive ones, $\boldsymbol{s}$, is very unlikely. Moreover, Figure 3.3 shows that the model misclassifies all transformed sections corresponding to $\boldsymbol{s}$ into the $\boldsymbol{n}$ and therefore the false-positive rate on *required* activities is very low. For instance, to see how RAE can establish a good utility privacy trade-off, consider the results for the Skoda dataset in Table 3.1(#1). We observe that the activity classifier can effectively recognize $\boldsymbol{r}$ activities (*e.g.* opening and closing doors), even when the app processes the output of RAE instead of the

Table 3.1: Results for the activity recognition on the Skoda dataset, using a pre-trained convolutional neural network [116]. $\boldsymbol{X}$ shows the original data, and $\boldsymbol{Y}$ shows the transformed data by RAE. Each number in the column "set of activities" refers to a corresponding activity label in Table 2.2.

		F1-score (%)	
#	*set of activities*	$\boldsymbol{X}$	$\boldsymbol{Y}$
1	$\boldsymbol{r} = \{2,3,5,6,7,9\}$	96.5	93.2
	$\boldsymbol{s} = \{4,8,10\}$	97.9	0.0
	$\boldsymbol{n} = \{0,1\}$	93.9	94.8
2	$\boldsymbol{r} = \{4,8,9,10\}$	97.9	96.3
	$\boldsymbol{s} = \{1,5,6,7\}$	96.2	0.0
	$\boldsymbol{n} = \{0,2,3\}$	94.3	93.4

		F1-score (%)	
#	set of activities	$\boldsymbol{X}$	$\boldsymbol{Y}$
3	$\boldsymbol{r} = \{2,3,5,6,7,9\}$	95.8	91.1
	$\boldsymbol{s} = \{4,8,10\}$	97.4	0.0
	$\boldsymbol{n} = \{0,1\}$	94.3	92.4
4	$\boldsymbol{r} = \{1,4,10\}$	97.6	95.0
	$\boldsymbol{s} = \{2,3,8,9\}$	98.0	0.0
	$\boldsymbol{n} = \{0,5,6,7\}$	92.3	88.2

Table 3.2: Results for the activity recognition on the the (left) Hand-Gesture and (right) the Opportunity dataset, using a pre-trained convolutional neural network [116]. $\boldsymbol{X}$ shows the original data, and $\boldsymbol{Y}$ shows the transformed data by RAE. Each number in the column "set of activities" refers to a corresponding activity label in Table 2.2.

		F1-score (%)	
#	set of activities	$\boldsymbol{X}$	$\boldsymbol{Y}$
1	$\boldsymbol{r} = \{1,2,3,4,9,10,11\}$	94.1	90.1
	$\boldsymbol{s} = \{5,6,7,8\}$	95.7	0.3
	$\boldsymbol{n} = \{0\}$	95.0	96.5
2	$\boldsymbol{r} = \{1,3,4,5,6,7\}$	95.2	90.4
	$\boldsymbol{s} = \{2,8,9,10,11\}$	94.5	0.6
	$\boldsymbol{n} = \{0\}$	95.0	97.5
3	$\boldsymbol{r} = \{1,3,4,5,6,7,8\}$	97.2	93.3
	$\boldsymbol{s} = \{2,9,10,11\}$	92.5	0.7
	$\boldsymbol{n} = \{0\}$	95.9	97.5
4	$\boldsymbol{r} = \{2,3,5,6,7,9\}$	96.1	92.1
	$\boldsymbol{s} = \{4,8,10\}$	97.0	0.5
	$\boldsymbol{n} = \{0,1\}$	95.7	97.6

		F1-score (%)	
#	set of activities	$\boldsymbol{X}$	$\boldsymbol{Y}$
1	$\boldsymbol{r}$={1,2,...,8,15,17}	76.9	75.9
	$\boldsymbol{s}$={9,10,...,14}	71.5	1.3
	$\boldsymbol{n}$={0,16}	84.4	82.1
2	$\boldsymbol{r}$={9,10,...,17}	71.8	64.3
	$\boldsymbol{s}$={1,2,...,8}	79.1	0.2
	$\boldsymbol{n}$={0}	88.9	89.7
3	$\boldsymbol{r}$={9,10,...,14,16}	74.9	77.1
	$\boldsymbol{s}$={1,2,3,4,15,17}	76.2	0.9
	$\boldsymbol{n}$ ={0,5,6,7,8}	85.0	81.6
4	$\boldsymbol{r}$={1,2,...,8,15,17}	70.3	65.0
	$\boldsymbol{s}$={9,10,...,14,16}	74.9	6.3
	$\boldsymbol{n}$={0,1}	93.7	92.9

raw data (with 93.2% accuracy). However, $\boldsymbol{s}$ activities (*e.g.* checking doors) that can be recognized with high accuracy when the app processes the raw data (with 97.9% accuracy), are completely filtered out in the output of the RAE. Moreover, the corresponding confusion matrix for this experiment in Figure 3.3 (Left-Bottom) shows that the utility of the *required* activities is preserved as the classifier wrongly infers all $\boldsymbol{s}$ activities as some $\boldsymbol{n}$ activities and not as one of $\boldsymbol{r}$ activities.

3.4.2 A Realistic Scenario

Considering the Utwente [99] dataset, let $\boldsymbol{s}$ = {typing, writing, smoking, eating} be the set of sensitive activities, $\boldsymbol{n}$ = {sitting, standing} be the set of neutral activities, and $\boldsymbol{r}$ = {walking, jogging, cycling, stairs-up, stairs-down} be the set of required activities. Considering a 2-second time window ($W = 100$ due to 50Hz sampling rate), we train an RAE with 6 hidden layers: 4 Convolutional-LSTM layers using *hyperbolic tangent* as the activation function with 256, 128,

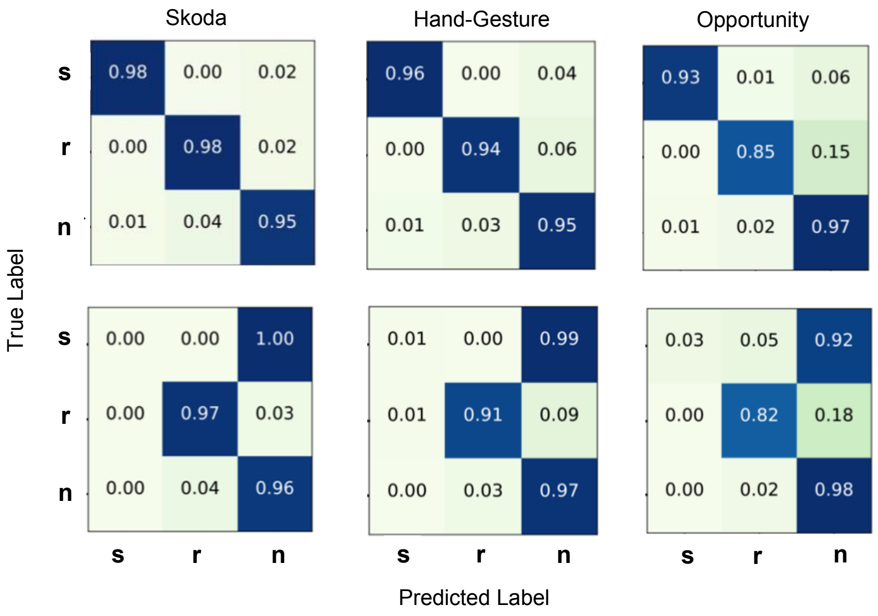


Figure 3.3: Confusion Matrix for (top) original time-series, and (bottom) transformed time-series by RAE. After transformation almost all the *sensitive* activities are recognized as *neutral* ones. The results correspond to case #1 of (left) Skoda dataset in Table 3.1, (middle) Hand-Gesture dataset in Table 3.2, and (right) Opportunity dataset in Table 3.2.

64, and 64 filters respectively, followed by 2 Convolutional layers using Scaled Exponential Linear Unit [190] as the activation function with 64 and 128 layers respectively. We also put a batch-normalizer [191] on the output of each hidden layer to reduce the training time.

To evaluate the privacy-utility trade-off, we use a DNN classifier. As we see in Table 3.3, the average accuracy of the classifier on the raw data is more than 99%. However, when we feed the same classifier with the output of the RAE, all the ***s*** activities are recognized as *sitting*, while the accuracy for ***r*** activities is almost equal to that of the raw data. As mentioned in Section 1.3 and Section 2.1.3, we emphasize that our assumption is that each time window only belongs to one activity of the user and we assume these time windows are sampled in an i.i.d. manner. To preserve privacy against an adversary that can take advantage of correlation among and timing of activities, each sensitive activity should be transformed into a neutral activity that does not leak information about performing a replacement at that time point. We discuss more on this in Chapter 6, as potential future work. We observe that for *smoking* there still is a 5% chance of recognition. Note that for some time windows of the *smoking* the raw data are similar to those of *standing* . This effect can be also seen in Table 3.3 (column *smoking*). This might be because of the labeling decision when the data curator labels intervals between cigarette drags as smoking behavior while the user is standing, but we lack proof of this due to the lack of this information in the dataset documentation from the original publishers.

Table 3.3: Confusion Matrix of the results on the test data for Utwente dataset. Rows show the true labels and columns show the predicted labels. In each cell, the left part shows the accuracy on the raw data, and the right part shows the accuracy after transformation. For brevity, all the values are rounded to one decimal point. Empty cells show 0.0 $\rightarrow$ 0.0.

	walking	**jogging**	**cycling**	**stairs-up**	**stairs-down**	**sitting**	**standing**	**typing**	**writing**	**eating**	**smoking**
walking	**97.5 $\rightarrow$ 97.2**			0.7 $\rightarrow$ 0.7	1.5 $\rightarrow$ 1.9						0.3 $\rightarrow$ 0.1
jogging		**100 $\rightarrow$ 100**									
cycling			**100 $\rightarrow$ 100**								
stairs-up	0.4 $\rightarrow$ 0.3	0.4 $\rightarrow$ 0.4	0.0 $\rightarrow$ 0.1	**98.8 $\rightarrow$ 98.8**			0.1 $\rightarrow$ 0.1				0.3 $\rightarrow$ 0.3
stairs-down				0.3 $\rightarrow$ 0.3	**99.7 $\rightarrow$ 99.7**						
sitting			0.0 $\rightarrow$ 0.3			**98.6 $\rightarrow$ 96.8**		1.0 $\rightarrow$ 0.0	0.1 $\rightarrow$ 0.0	0.1 $\rightarrow$ 0.0	0.1 $\rightarrow$ 2.8
standing			0.0 $\rightarrow$ 0.3				**99.4 $\rightarrow$ 98.2**				0.6 $\rightarrow$ 1.5
typing						**0.0 $\rightarrow$ 100**		**100 $\rightarrow$ 0.0**			
writing			0.0 $\rightarrow$ 0.7			**0.0 $\rightarrow$ 99.3**			**99.9 $\rightarrow$ 0.0**	0.1 $\rightarrow$ 0.0	
eating			0.0 $\rightarrow$ 0.5			**0.1 $\rightarrow$ 99.4**	0.3 $\rightarrow$ 0.0			**99.6 $\rightarrow$ 0.0**	0.1 $\rightarrow$ 0.1
smoking			0.0 $\rightarrow$ 0.1			**0.0 $\rightarrow$ 94.9**				2.3 $\rightarrow$ 0.0	**97.5 $\rightarrow$ 5.0**

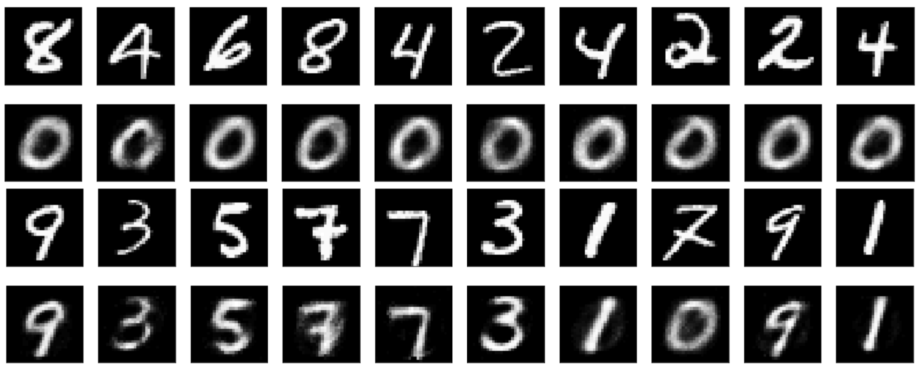


Figure 3.4: An example of RAE's performance on the MNIST dataset. We consider 0 as the neutral digit, other even numbers as the sensitive digits, and the odd digits as the required digits. The first and third rows show the original digit, while the second and fourth rows show the corresponding transformation done by the trained RAE.

3.4.3 Visualization

It is a very challenging task to visualize multi-channel time-series, especially in our multi-dimensional setting. In order to clarify the idea of RAE, we present and discuss some perceptible examples.

First, we consider the MNIST dataset of handwritten digits [192]. It contains $60,000$ training images and $10,000$ testing images. As an explicit example, in our setting, we consider 0 as a kind of neutral information, other even numbers $(2, 4, 6, 8)$ as a set of sensitive information, and all of the odd numbers $(1, 3, 5, 7, 9)$ as a set of the required information we want to keep unchanged. Figure 3.4 shows the output of the RAE when it receives some samples from the test set as input. The top part of Figure 3.4 shows that RAE transforms all of the images categorized as sensitive information into images that are visually very similar to the digit 0 which is considered as neutral, whereas the bottom part of Figure 3.4 shows that the required information is kept almost unchanged; except for one case that digit 7 is wrongly transformed into 0. This is just a simple example to build an intuitive sense of how our method can learn the piecewise function described in equation 3.3.1 to automatically replace sensitive information in personal data.

Second, as another visualization example, we use t-Distributed Stochastic Neighbor Embedding (t-SNE) [193] which is a well-known algorithm for 2D visualization of high-dimensional data. The t-SNE is in fact a technique for dimensionality reduction and as well it is a known strategy for showing the similarity relationships among different classes in a dataset. It uses PCA to compress high-dimensional data into a lower-dimensional space, then maps the

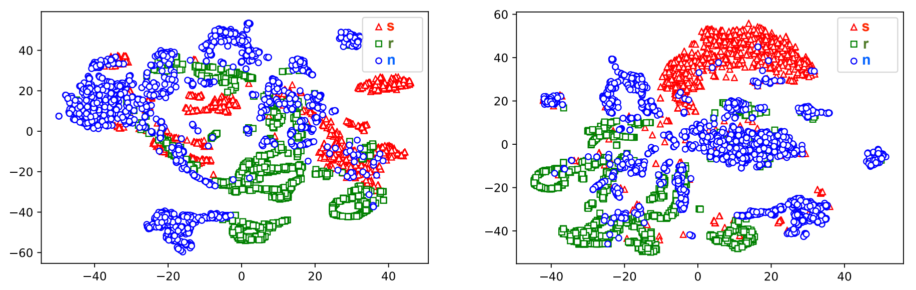


Figure 3.5: 2D visualization of the sensitive $\boldsymbol{s}$, required $\boldsymbol{r}$, and neutral $\boldsymbol{n}$ activities in the Skoda dataset (#1 in table 3.1) using t-SNE: (left) original data $\boldsymbol{X}$, and (right) transformed data $\boldsymbol{Y}$.

compressed data to a 2D plane that can be plotted easily. Since t-SNE aims to preserve the local structure of the high-dimensional data, we use it to describe how the RAE would push sensitive data points, in the original data space, into areas that correspond to neutral data, and at the same time preserve the separability and structures of the required data points.

Figure 3.5 shows the similarity relationships among different classes of activities for Skoda dataset. Regarding the scatter plots depicted in Figure 3.5, sensitive data points in the original time-series (left plot) have their own structures and clearly separable from other classes. But, in the transformed version of time-series (right plot) almost all sensitive data points have been moved into another area of the data space which is more closely related to neutral data points. They also lose their recognizable patterns and all of them mapped to almost the same area of the space. On the other hand, RAE maintains particular patterns of required data points, thus causes no harm to the recognition accuracy of desired activities. Note that in this example, we reduce the dimensionality from 1800-d to just 2-d, which leads to huge information loss. t-SNE has two tunable parameters, *initial dimension* (here is set to 100), and *perplexity* (here is set to 60) that is defined as 2 to the power of the Shannon entropy.

3.5 A Potential Attack

We assume that an adversary has access to the RAE that is used by the users to transform their data (*i.e.* white-box access). Moreover, the adversary has access to the user's transformed data, and a dataset of the user's sensor data. We consider a setting where, observing a time window released by the user that is classified as one of the neutral activities, the adversary aims to determine whether this time window is *real* (*i.e.* the time window really corresponds to the classified neutral activity) or *fake* (*i.e.* it was originally a sensitive data and it is replaced). Notice that the adversary recognizes each time window as either required or neutral, as all sensitive time windows are transformed to neutral ones. Therefore, if the adversary can distinguish between *real* and *fake* neutral time window, then it can make sure that the fake ones are the result of a transformation of a sensitive activity. Thus, in such case the RAE is no more useful than a filtering approach that just does not release sensitive time windows at all. Hence it is important to know that to what extent the replacement of sensitive data with neutral data is indistinguishable from the real neutral data.

For this purpose, we use a Generative Adversarial Network (GAN) [114]. A GAN is a DNN that learns to generate fake data that are very similar to the samples drawn from the true data distribution. GANs consist of two models: a *generator* and a *discriminator*. The generator takes random input values and uses feedback from the discriminator to produce convincing data that the discriminator cannot distinguish from real data. The discriminator, which is usually a binary classifier, determines whether a given input looks like a real input from the dataset or like artificially generated data.

We implement the GAN proposed in [194] for evaluating the RAE as follows. First, we assume that the adversary trains a GAN on the available dataset, such that the generator learns to produce time windows very similar to real neutral time windows and the discriminator learns to determine whether the given time windows, recognized as neutral data, are real or fake one. Second, after training the GAN, we separate the discriminator and give it as input: (i) *real* neutral time windows, (ii) *fake* neutral time windows, (iii)*randomly generated* neutral time windows by the generator, and (iv) *the top 10% randomly generated* neutral time windows as rated by the discriminator. We measure the binary classification accuracy rate for these four categories. The results, shown in Figure 3.6, demonstrate that the output of the RAE is almost as similar as the random data generated by the generator but not quite as good as the best data generated by the generator. We see that when we give the discriminator more time to learn, it eventually distinguishes between real neutral data and fake ones. Therefore, if an adversary can get access to some of the user's sensor data, the adversary can use GAN to distinguish between real and fake non-sensitive data. Thus, in this situation, the safety of our proposed replacement method will be reduced to the safety of the filtering approach.

We also conduct another experiment to investigate whether the privacy of user i can be compromised by having access to the original data of another user j. Figure 3.7 shows the accuracy of the discriminator for distinguishing between real and fake neutral data of subject #3 in the Opportunity dataset when the discriminator has been trained on neutral data from subject #1. Results show

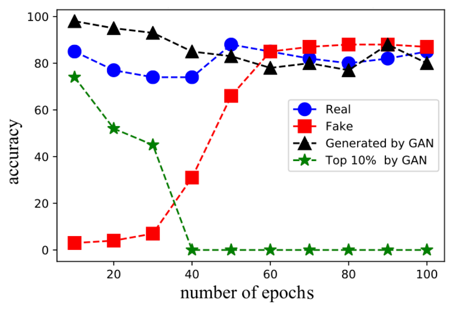


Figure 3.6: The accuracy of the GAN's discriminator in distinguishing different kinds of real, fake, and generated time windows on the Skoda dataset (#1 in table 3.1).

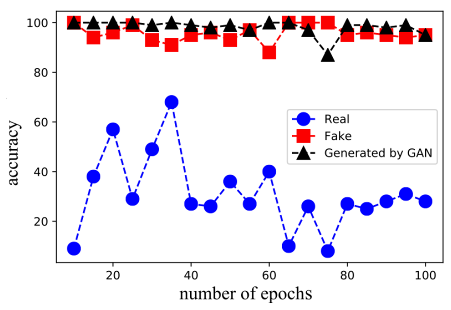


Figure 3.7: The accuracy of the GAN's discriminator in detecting real neutral data by having access to other users' data on the Opportunity dataset (#1 in table 3.2).

that the discriminator cannot recognize real neutral data as real, and its error rate is about 70% when it converges. Although the adversary recognizes all fake neutral data as fake, it is not valuable because it also recognizes real data as fake. Note that in this experiment the list of sensitive and neutral inferences is the same for both data subjects. Moreover, we can see from Figure 3.6 that the adversary's accuracy begins to improve slowly from epoch 10 to epoch 30 and stabilized by epoch 70. By contrast, without access to the data of the target user, even after 100 epochs in Figure 3.7 there is no consistent pattern of improvement.

3.6 Summary

We showed that in dealing with sensitive patterns in sensor time series, a *replacement* algorithm, addresses some important challenges. First, it is rational to say that not only the recognition of a specific sensitive activity, but also the detection of the occurrence of any kind of sensitive activities can violate a user's privacy. Perturbation by adding noise to a sensitive time window or completely filtering those time windows, although avoid the explicit recognition of the exact activity, yet clearly shows that something sensitive has happened at that moment. An additional knowledge that can be used as side-channel information for other potential attacks on the user's privacy. Moreover, perturbation algorithms not only need to distinguish sensitive and non-sensitive time windows from each other, but also to compute a correlated noise based on the structure of the current time window such that it cannot be easily filtered out.

A *replacement* algorithm addresses this concern by deceiving the app into falsely believing that another activity is happening. For example, during walking in a park, the user stops to smoke. Here, replacing patterns related to the smoking activity with some patterns related to the sitting activity can cover sensitive behavior. Second, transformation algorithms that map each time window into a lower-dimensional feature vector, or any other representation, restricts the app's functionality into using classifiers that are consistent with that representation. A *replacement* algorithm keeps the data dimensions unchanged without enforcing any limitations on the model that they can choose. Despite these advantages, the replacement approach is still susceptible to adversaries with some other background knowledge, similar to the attack we showed using GANs. Other potential attacks can also happen by taking advantage of the likelihood of each activity at a specific time or the temporal correlation among different activities. For example, observing 2 minutes *sitting* right after observing *going downstairs* activity and right before recognizing *going upstairs* activity can leak some information that such *sitting* might be a replacement of a *smoking* activity, because of the temporal correlation with other observed activities. We discuss more on this in Chapter 6 and leave it as a potential future work.

Chapter 4

Sensor Data Anonymization

4.1 Overview

The RAE algorithm (presented in Chapter 3) aims to protect a user's sensitive activities, but the range of sensitive information can go beyond just a subset of our daily activities and may contain other types of information that might be discovered from data patterns observed in our daily activities. Motion patterns can facilitate the re-identification of users [45], or reveal some personal attributes, such as gender [195, 44]. When the user's sensor data are collected, for example for research purposes [196, 197], users may want to minimize the risk of being re-identified by those who will access the collected data, and researchers may want to infer the user's activities. Although there are many ways to identify a user in a dataset, here re-identification based on sensor data pattern refers to analyzing a sample time series of sensor data and figure out whether that data belongs to a specific user or not, possibly by taking advantage of some data of that user collected through another channel. To cover such privacy concerns, we need a data transformation algorithm to protect such sensitive information that can be discovered from all activities and helps an adversary in re-identification purposes.

In this chapter, we propose Anonymizing AutoEncoder (AAE) to transform sensor data to prevent the exposure of sensitive information that are user-specific (*e.g.* gender information), while simultaneously preserving some information in the data that help us to recognize the user's activities (see Figure 4.1). We formulate the problem into an information-theoretic framework [71, 164], then propose a multi-objective loss function that helps AAE to minimize the amount of sensitive information while keeping the data utility regarding the desired task. This is achieved by forcing the encoder of AAE to produce an uninformative latent representation as well as forcing the decoder of AAE to

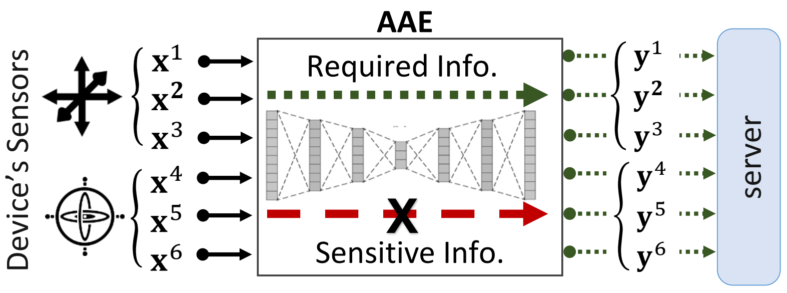


Figure 4.1: Overview of the AAE. For each sensor time window, every stream of sensor data, $\mathbf{x}^i$, is transformed into a stream with the same length, $\mathbf{y}^i$, then is shared with the server. The transformation aims to keep the pre-specified required information, but preventing the pre-specified sensitive information to be inferred.

reconstruct data such that it cannot be confidently assigned to a specific user in the training set. Moreover, by imposing a constraint on the amount of distortion, AAE tries to reconstruct data such that it is useful for an activity recognition app.

It is worth noting that in the problem formulation of Chapter 3, each sensor time window is mapped to only one categorical variable, *i.e.* the type of user's activity. Therefore, in RAE, we assumed that some values of that variable (*i.e.* label) indicate sensitive activities and other values indicate non-sensitive ones. However, in the problem formulation of this chapter, a sensor time window is mapped to two categorical variables, *i.e.* the activity and the attribute of the user. Here, for AAE, all values of the activity variable are assumed as non-sensitive, and all values of the attribute variable are assumed as sensitive. Therefore, in this chapter, we show how to formulate the problem as an information-theoretic minimax game that aims to keep one variable and hide another one. But for RAE, it was not straightforward to design such a minimax game, as we needed to keep some activities while replacing some others at the same time.

We evaluate AAE on two datasets of human activity recognition using state-of-the-art deep neural networks as the classifier. We show that the AAE can preserve the usefulness of the transformed data for activity recognition similar to that of the original data, while reducing the accuracy of gender and identity recognition close to the random guess in the corresponding dataset[1].

[1]Code and data to reproduce results are publicly available at `https://github.com/mmalekzadeh/motion-sense`

4.2 AAE Methodology

Let $\mathbf{r} \in \{0,1\}^R$ and $\mathbf{s} \in \{0,1\}^S$ be one-hot vectors representing the *required* (*e.g.* activity) and *sensitive* (*e.g.* gender or identity) data, respectively. Let $\hat{\mathcal{M}}(\cdot)$ be a data transformation algorithm, $\mathbf{X}$ be the data want to anonymize, and $\mathbf{Y} = \hat{\mathcal{M}}(\mathbf{X})$ be the transformed data. We define the fitness function $\mathcal{F}(.)$ as

$$\mathcal{F}\Big(\hat{\mathcal{M}}(\mathbf{X})\Big) = \beta_s \mathtt{I}\Big(\mathbf{s}; \hat{\mathcal{M}}(\mathbf{X})\Big) - \beta_r \mathtt{I}\Big(\mathbf{r}; \hat{\mathcal{M}}(\mathbf{X})\Big) + \beta_d \mathcal{D}\Big(\mathbf{X}, \hat{\mathcal{M}}(\mathbf{X})\Big), \tag{4.2.1}$$

where $\mathtt{I}(\cdot,\cdot)$ is the mutual information, $\mathcal{D}(\cdot,\cdot)$ is a distance metric (*e.g.* Mean Squared Error), and the non-negative, real-valued weights β_s, β_r and β_d are the parameters[2]. that can be used for making a trade-off between *privacy* and *utility* (as formulated in Section 1.3). Let the optimal anonymization function, $\mathcal{M}(\cdot)$, that transforms $\mathbf{X}$ into $\mathbf{Y}$ be defined as

$$\mathcal{M}(\mathbf{X}) = \underset{\hat{\mathcal{M}}(\mathbf{X})}{\arg\min}\, \mathcal{F}\Big(\hat{\mathcal{M}}(\mathbf{X})\Big). \tag{4.2.2}$$

The threefold objective of Equation (4.2.1) is to minimize $\mathtt{I}\Big(\mathbf{s}; \hat{\mathcal{M}}(\mathbf{X})\Big)$, the mutual information between the sensitive data and the transformed data; to maximize $\mathtt{I}\Big(\mathbf{r}; \hat{\mathcal{M}}(\mathbf{X})\Big)$, the mutual information between the required data and the transformed data (*i.e.* to minimize its negative value); and, to avoid large distortions in the data by minimizing $\mathcal{D}\Big(\mathbf{X}, \hat{\mathcal{M}}(\mathbf{X})\Big)$, the distance between the original data and its transformation.

As we cannot practically search over all possible transformation functions, we consider a DNN and look for the optimal parameter set through training. To approximate the required mutual information terms, we reformulate the optimization problem in Equation (4.2.2) as a DNN optimization problem. Let $\hat{\mathcal{M}}(\mathbf{X};\theta)$ be a DNN, where θ is the parameter set of the DNN. Ideally, the training procedure aims to find the optimal parameter set θ^* by searching the space of all possible parameter sets, Θ, as:

$$\theta^* = \underset{\theta \in \Theta}{\arg\min}\, \beta_s \mathtt{I}\Big(\mathbf{s}; \hat{\mathcal{M}}(\mathbf{X};\theta)\Big) - \beta_r \mathtt{I}\Big(\mathbf{r}; \hat{\mathcal{M}}(\mathbf{X};\theta)\Big) + \beta_d \mathcal{D}\Big(\mathbf{X}, \hat{\mathcal{M}}(\mathbf{X};\theta)\Big). \tag{4.2.3}$$

We define $\mathcal{M}(\cdot;\theta^*)$ as the best transformation algorithm for a general $\mathcal{M}(\cdot)$ in Equation (4.2.2), realized by a DNN. It should be noted that, in practice, using off-the-shelf DNN optimizers for a non-convex loss function, we cannot guarantee to find the optimal parameter set (*i.e.* the best transformation algorithm), but to find a near-optimal parameter set. Moreover, in practice we cannot compute the *exact* value of mutual information, $\mathtt{I}(\cdot;\cdot)$. However, an *estimation* of mutual

[2]Note that here we use three β parameters for the ease of exposition, but two would be enough as theoretically it is only the ratios that matter

information can be achieved via log-likelihood estimation using an adversarial approximation [147, 88]. In Section 4.3, we elaborate more on how to estimate $\mathtt{I}(\cdot;\cdot)$ using adversarial training via neural networks.

Finally, similar to the setting of previous work [86, 88], a summary of our assumptions on the provided utility and privacy to the user by AAE is:

- *Utility.* Each time window of sensor data is transformed such that an activity recognition app, simulated via a parameterized DNN, can recognize the user's activity by processing the transformed data, with an accuracy close to what it could have got if we have released the original data (*i.e.* high classification accuracy for activity recognition task).

- *Privacy.* First, each time window is transformed such that an app, again simulated via a parameterized DNN, cannot recognize the identity/gender of the user from the transformed data (*i.e.* very low classification accuracy for identity/gender recognition task). Second, following the notion of inferential privacy, the empirical estimation of mutual information is achieved by an adversarial training of a classifier using log-likelihood loss function (see Section 2.2.3).

- *Adversary.* First, similar to Chapter 3, we assume each *time-window* only contain patterns that correspond to one activity, and we assume that each time window is sampled i.i.d. and thus we do not address situations where there might be any temporal or long term correlation between different time windows. Second, we assume that adversaries only have access to the data released by the AAE to launch an attack, and do not consider adversaries that can take advantage of other side information that they can collect about users. Third, we assume that adversaries are computationally bounded and use empirical machine learning models to perform the classification of the user's data.

4.2.1 Multi-Objective Loss Function

The key contributor to the AAE training is the multi-objective loss function

$$\mathcal{L} = \beta_s \mathcal{L}_s + \beta_r \mathcal{L}_r + \beta_d \mathcal{L}_d, \tag{4.2.4}$$

where $\mathcal{L}_r$ and $\mathcal{L}_d$ are the *utility losses* that can be customized based on the target task requirements, whereas $\mathcal{L}_s$ is the *privacy loss* that helps the AAE to remove data patterns that facilitate the inference of sensitive information. This loss function is used by the AAE optimizer searching for the θ^* in Equation 4.2.3.

Practically, the categorical cross-entropy loss function for classification, $\mathcal{L}_r = -\mathbf{r}\log(\hat{\mathbf{r}})$, can be used to preserve the required information, where $\hat{\mathbf{r}}$, the output of a softmax function, is an R-dimensional vector that sums to 1 and is supposed to simulate the conditional probability of the label given the observed data. The distance function that controls the amount of distortion, $\mathcal{L}_d$, forces $\mathbf{Y}$ to be as similar as possible to the input $\mathbf{X}$:

$$\mathcal{L}_d = \frac{1}{W \times H} \sum_{w=1}^{W} \sum_{h=1}^{H} (x_{wh} - y_{wh})^2. \tag{4.2.5}$$

Finally, the privacy loss, $\mathcal{L}_s$, the most important term of our multi-objective loss function that aims to minimize sensitive information in the data, is defined as:

$$\mathcal{L}_s = -\Big(\mathbf{s} \cdot \log(\mathbf{1}^S - \hat{\mathbf{s}}) + \log\Big(1 - \max(\hat{\mathbf{s}})\Big)\Big), \tag{4.2.6}$$

where $\mathbf{1}^S$ is the all-one column vector of length S, $\mathbf{s}$ is the ground truth one-hot[3] vector for $\mathbf{X}$, and $\hat{\boldsymbol{s}}$ is the output of the classifier's softmax layer that is an S-dimensional real-valued column vector, and $\cdot$ denotes the dot product of two column vectors. The first term, $\mathbf{s} \cdot \log(\mathbf{1}^S - \hat{\mathbf{s}})$ penalizes AAE if the classifier's prediction for the true identity in is close to 1, and the second term, $1 - \max(\hat{\mathbf{s}})$, penalizes AAE if the classifier is also too certain about any other (wrong) identity label.

The goal of training AAE is to minimize the *privacy loss* by minimizing the amount of information leakage from $\mathbf{s}$ to $\mathbf{Y}$, and we show, in Section 4.2.2, how our loss function $\mathcal{L}_s$ can achieve this. A trivial transformation would consistently transform data of $\mathbf{s}_i$ into the data of $\mathbf{s}_j$ (and vice versa). However, this transformation would only satisfy the first element of $\mathcal{L}_s$. As no adversary should be able to confidently predict $\mathbf{s}$ from $\mathbf{X}$, we maximize the difference between the prediction, $\hat{\mathbf{s}}$, and the ground truth, $\boldsymbol{s}$ by minimizing the cross-entropy between these two values, as well as the maximum value of the predicted vector $\hat{\mathbf{s}}$. Although the $\max(\cdot)$ function is not differentiable, it is piecewise differentiable and in practice it can also be implemented using the corresponding soft version of the maximum function that is $(\text{softmax}(z))^{\mathsf{T}} \cdot z$ [198].

4.2.2 Derivation of the Privacy Loss

The main goal of our AAE is to make the hidden sensitive factor $\mathbf{s}$ and the released data $\mathbf{Y}$ as independent of each other as possible. To this end, we minimize the amount of information leakage from $\mathbf{s}$ to $\mathbf{Y}$. As no function $\mathcal{G}$ can increase the available information in the data [72], the inequality $\mathtt{I}(\mathbf{s}; \mathbf{Y}) \geq \mathtt{I}(\mathbf{s}; \mathcal{G}(\mathbf{Y}))$,

[3] A column vector with zeros except for a 1 in the row corresponding to the correct class.

holds, and therefore if we reduce the mutual information between $\mathbf{s}$ and $\mathbf{Y}$, we technically put an upper bound on the amount of sensitive information leakage through sharing $\mathbf{Y}$.

We know that $\mathtt{I}(\mathbf{s};\mathbf{Y}) = \mathtt{H}(\mathbf{s}) + \mathtt{H}(\mathbf{Y}) - \mathtt{H}(\mathbf{s},\mathbf{Y})$, and $\mathtt{H}(\mathbf{s})$ is fixed as it does not depend on our transformation function. Thus, to minimized $\mathtt{I}(\mathbf{s};\mathbf{Y})$, a transformation should (i) minimizes $\mathtt{H}(\mathbf{Y})$, and (ii) maximizes the $\mathtt{H}(\mathbf{s},\mathbf{Y})$. For example, authors in [74] show that the downsampling of sensor data makes re-identification of users much harder, as it directly reduces $\mathtt{H}(\mathbf{Y})$. We will show a similar result in Figure 4.5. However, reducing $\mathtt{H}(\mathbf{Y})$ independent of the sensitive variables, *e.g.* by downsampling to a very low sampling rate, can lead to a substantial utility loss [74]. Instead, we can focus on maximizing $\mathtt{H}(\mathbf{s},\mathbf{Y})$. In other words, instead of blindly reducing as much information as we can from (*i.e.* by decreasing the entropy $\mathtt{H}(\mathbf{Y})$), we can focus on only reducing the sensitive information that we do not want to share (*i.e.* by increasing the joint entropy $\mathtt{H}(\mathbf{s},\mathbf{Y})$).

For practical purposes, we need an estimator for $\mathtt{H}(\mathbf{s},\mathbf{Y})$ as we cannot calculate the joint entropy directly for high-dimensional data. When labeled data are available, $\mathbf{Y}$ can be used as input to predict $\hat{\mathbf{s}}$ as an estimation of $\mathbf{s}$. We therefore reformulate the problem of maximizing the joint entropy, $\mathtt{H}(\mathbf{s},\mathbf{Y})$, as maximization of the cross-entropy between the true label, $\mathbf{s}$, and the predicted label, $\hat{\mathbf{s}}$: $\mathtt{H}_{\hat{\mathbf{s}}}(\mathbf{s}) = -\sum_{i=1}^{S}\mathbf{s}_i \log \hat{\mathbf{s}}_i$, where $\hat{\mathbf{s}}_i$ is the i-th element of the vector predicted by the multi-class classifier. The empirical cross-entropy between $\mathbf{Y}$ of a ground truth sensitive label $\mathbf{s}$ with $\mathbf{s}_i = 1$ is $-\mathbf{s}\log\hat{\mathbf{s}} = -\log\hat{\mathbf{s}}_i$ and, since $\hat{\mathbf{s}}_i \in (0,1]$, maximizing $-\log\hat{\mathbf{s}}[k]$ is equivalent to minimizing $-\log(1-\hat{\mathbf{s}}[k])$.

Therefore, minimizing the first term of Equation (4.2.6), $\mathbf{s}\log(\mathbf{1}^S - \hat{\mathbf{s}})$, helps us to empirically minimize the mutual information, $\mathtt{I}(\mathbf{s};\mathbf{Y})$, and, by forcing the AAE to minimize this value, we minimize the amount of privacy loss. We discussed this notion of inferential privacy in Chapter 2, Section 2.2.3, and encourage readers to find more rigorous analyses and applications of inferential privacy approach for minimizing the mutual information in [71, 147, 85, 199, 88].

4.3 AAE Training Procedure

Figure 4.2 shows the framework for the training of an AAE. The Encoder maps a received time window, $\boldsymbol{X} \in \mathbb{R}^{W\times H}$, into a lower-dimensional latent vector $\boldsymbol{z}$. The Decoder gets $\boldsymbol{z}$ and produces a reconstruction, $\boldsymbol{Y}$, of the original time window. The Encoder Classifier and the Decoder Classifier process $\boldsymbol{z}$ and $\boldsymbol{Y}$, respectively, to measure how they are informative about the sensitive data $\mathbf{s}$. The Required Data Classifier performs a similar task for measuring how $\boldsymbol{Y}$ is informative about the required data $\mathbf{r}$. Finally, the distortion measurement, a

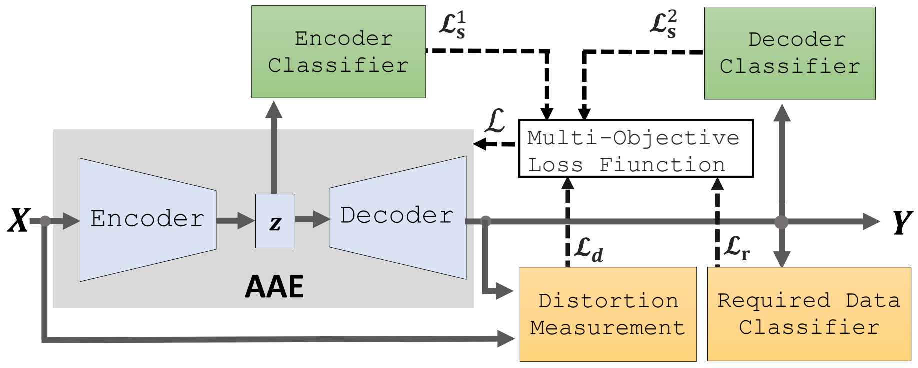


Figure 4.2: The losses involved in the training procedure. After training, the Anonymizing AutoEncoder (AAE). Solid lines show data flow; dashed lines show loss functions feedback. $\boldsymbol{X}$ is the original time window and $\boldsymbol{z}$ is the lower-dimensional representation of the input data. $\boldsymbol{Y}$ is the transformed time window. $\mathcal{L}_s$ is the privacy loss, $\mathcal{L}_r$ is the utility loss corresponding to the required data, and $\mathcal{L}_d$ is the distortion loss function. $\mathcal{L}$ is the overall loss function for training the AAE in Equation 4.2.4.

Algorithm 4.1 Training AAE

```
1: Input: $\mathbb{D}$: training datasets, $\mathbb{G}^{\mathbf{s}}$: ground truth labels for sensitive data, $\mathbb{G}^{\mathbf{r}}$: ground truth labels of required data, $E$: number of epochs, $\mathcal{C}^E$: Encoder Classifier, $\mathcal{C}^D$: Decoder Classifier, $\mathcal{C}^R$: Required Data Classifier, $\mathcal{M}^E$: Encoder part of the AAE, $\mathcal{M}^D$: Decoder part of the AAE, $(\beta_s, \beta_r, \beta_d)$: parameters for a desired trade-off.
2: Output: $\theta$: the AAE's optimized parameters.
3: for $e = 1$ to $E$ do
4:   for $(\boldsymbol{X}, \mathbf{s}, \mathbf{r})$ in $(\mathbb{D}, \mathbb{G}^{\mathbf{s}}, \mathbb{G}^{\mathbf{r}})$ do
5:     $\boldsymbol{z} = \mathcal{M}^E(\boldsymbol{X})$
6:     $\boldsymbol{Y} = \mathcal{M}^D(\boldsymbol{z})$
7:     $\mathcal{L}_s = \text{ClassificationLoss}(\mathcal{C}^E(\boldsymbol{z}), \mathbf{s}) + \text{ClassificationLoss}(\mathcal{C}^D(\boldsymbol{Y}), \mathbf{s})$
8:     $\mathcal{L}_r = \text{ClassificationLoss}(\mathcal{C}^R(\boldsymbol{Y}), \mathbf{r})$
9:     $\mathcal{L}_d = \text{DistortionLoss}(\boldsymbol{X}, \boldsymbol{Y})$
10:    $\mathcal{L} = \beta_s\mathcal{L}_s - \beta_r\mathcal{L}_r + \beta_d\mathcal{L}_d$
11:    $\mathcal{G} = \text{Backpropagation}(\mathcal{L})$
12:    $\theta = \text{optimize}(\theta, \mathcal{G})$
13:  end for
14: end for
```

loss function that constrains the allowed distortion on the data, gets the original data, $\boldsymbol{X}$, and reconstructed data, $\boldsymbol{Y}$, to quantify the amount of distortion that is done, for example, by using MSE.

First, we train all the classifiers on the training dataset for several epochs, following the typical procedures for training DNNs. Then, using a similar idea of adversarial training [114], we train AAE using the procedure that is explained in Algorithm 4.1. The trained classifiers help to approximate the amount of information that is carried by the transformed data. In fact, unlike other adversarial methods [114, 200], our objective is not to learn the data distribution, but to transform a time window, that is informative about users' sensitive data, into another time window which (ideally) carry only information about the required data, with minimum possible distortion. This training procedure can be done through a trusted mediator, or it can be done locally on the user side, or a user can download a public pre-trained model and refined it on their own data [201].

4.4 Evaluation

4.4.1 Datasets

To evaluate the AAE, we use my contributed MotionSense dataset and the MobiAct [175] dataset (see Table 2.2).

4.4.2 Protecting the User's Identity

To evaluate the AAE, we first measure the extent to which the accuracy of activity recognition suffers from anonymization compared to accessing the original data. We measure the trade-off between recognizing users' activities (required information) and their identity (sensitive information). We compare AAE with baseline methods for coarse-graining time-series data (*resampling* and *singular spectrum analysis*) and with the method in [86] that only considers the information included in latent representation, $\boldsymbol{z}$, without taking the reconstructed data, $\boldsymbol{Y}$, into account.

We use MotionSense[4] and consider two methods of dividing the dataset into training and test sets, namely *Subject* and *Trial*. For *Subject*, we put all data of 4 of the users in the dataset, 2 females and 2 males, as testing data and the remaining 20 users as training. We use the *Subject* setting for evaluating the activity recognition task, thus the accuracy of activity recognition is evaluated on data of 4 users that are not used during the training. As using the *Subject* setting is not meaningful for the identity recognition task, we consider the *Trial* setting where we put one trial data of each user as testing data and the remaining trials of that user's data as training. For example, as we have three walking trials for every user, we consider one trial as testing and the other two as training. We use the *Trial* setting for the identity recognition task. In both cases, we put 20% of the training data for validation during the training phase. We repeat each experiment 5 times and report the mean and the standard deviation. For

[4]The majority of public datasets of motion sensor data do not simultaneously satisfy the requirements of abundance and variety of activities and users, and datasets that satisfy both are not publicly available (e.g. [197, 45].)

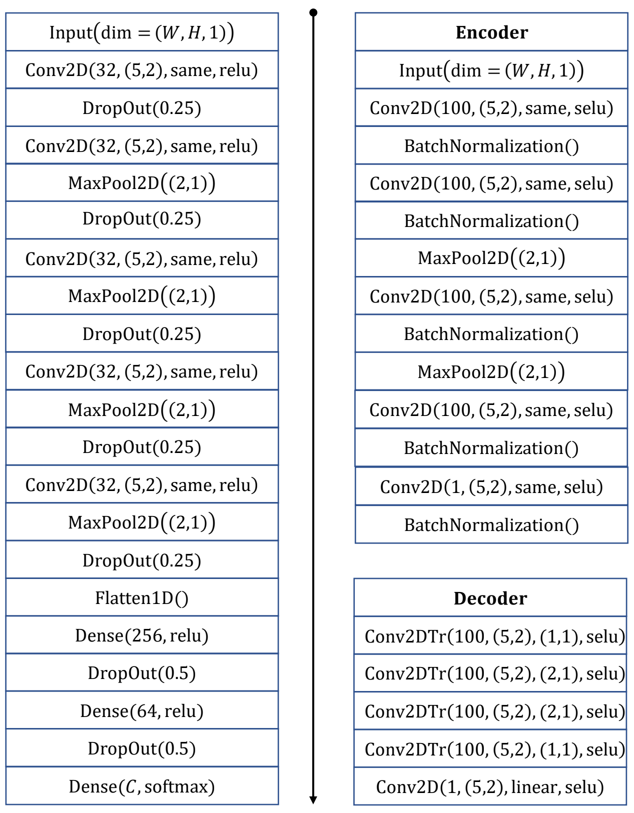

Input(dim = (W, H, 1))
Conv2D(32, (5,2), same, relu)
DropOut(0.25)
Conv2D(32, (5,2), same, relu)
MaxPool2D((2,1))
DropOut(0.25)
Conv2D(32, (5,2), same, relu)
MaxPool2D((2,1))
DropOut(0.25)
Conv2D(32, (5,2), same, relu)
MaxPool2D((2,1))
DropOut(0.25)
Conv2D(32, (5,2), same, relu)
MaxPool2D((2,1))
DropOut(0.25)
Flatten1D()
Dense(256, relu)
DropOut(0.5)
Dense(64, relu)
DropOut(0.5)
Dense(C, softmax)
Encoder
Input(dim = (W, H, 1))
Conv2D(100, (5,2), same, selu)
BatchNormalization()
Conv2D(100, (5,2), same, selu)
BatchNormalization()
MaxPool2D((2,1))
Conv2D(100, (5,2), same, selu)
BatchNormalization()
MaxPool2D((2,1))
Conv2D(100, (5,2), same, selu)
BatchNormalization()
Conv2D(1, (5,2), same, selu)
BatchNormalization()
Decoder
Conv2DTr(100, (5,2), (1,1), selu)
Conv2DTr(100, (5,2), (2,1), selu)
Conv2DTr(100, (5,2), (2,1), selu)
Conv2DTr(100, (5,2), (1,1), selu)
Conv2D(1, (5,2), linear, selu)

Figure 4.3: The DNN architecture used to evaluate AAE. (left) DNN used for building all the classifiers in Figure 4.2. The only differences among these classifiers are that W for Encoder Classifier is 32, whereas for Decoder and Required Data classifiers is 128, and C for Required Data classifier is 4, whereas for the other two is 24 as we have 24 users and 4 activities. (right) The architecture of Encoder and Decoder. For the details, we refer to TensorFlow's naming conventions [202].

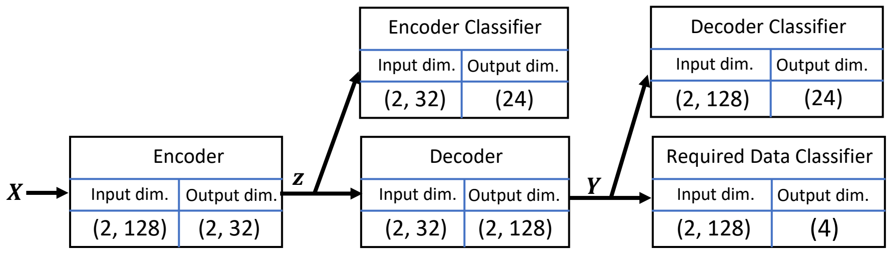


Figure 4.4: Data flow during the training of the AAE.

all the experiments we use the magnitude value for both the gyroscope and accelerometer.

For all the classifiers, we use the DNN architecture depicted in Figure 4.3 (left) using categorical cross-entropy loss function [198]. To prevent overfitting

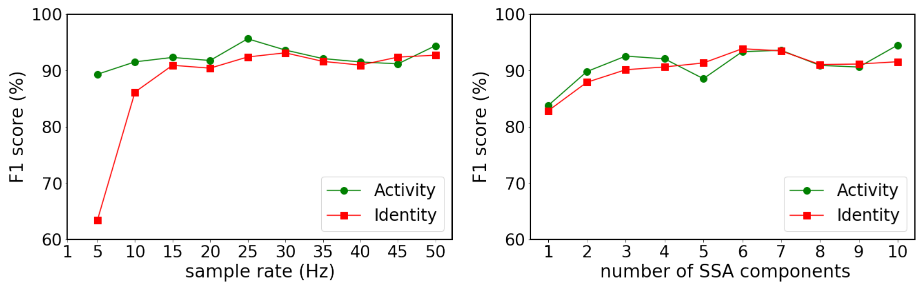


Figure 4.5: Classification accuracy for a deep convolutional neural network for both Activity and Identity recognition. (eft) Using data resampled to another rate (from 5 to 50 Hz, where 50 Hz is the original sampling rate). (right) Using data reconstructed using only a subset of components (from 1 to 10, from a total of 50), ordered from largest to smallest by corresponding singular values.

to the training data, we put a Dropout [203] layer after each convolutional layer. We also use L2 regularization to penalize large weights so that the classifier is forced to learn features that are more relevant for the prediction. Similarly, the DNN architecture used as AAE is depicted in Figure 4.3 (right). Figure 4.4 shows the overall architecture. To simplify the process of encoding data into a lower-dimensional representation and then decoding it to the original dimension with convolutional filters, we set W to be a power of 2. The larger length of time window W, the lower the possibility of taking advantage of the correlation among the successive windows by adversaries. But larger window sizes increase the delay for real-time apps. We set $W = 128$ (*i.e.* 2.56 seconds), following the setting used in state-of-the-art papers for activity recognition [101, 103, 102].

4.4.2.1 Baseline Methods

Theoretically speaking, downsampling can reduce the amount of information included in the data [72]. One can use downsampling to reduce the richness of the data to the extent that it contains the minimum acceptable information for recognizing the user's activity but not more fine-grained information that reveals other user's sensitive attributes. Figure 4.5 (left) 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 rates greater than 20 Hz. However, even at 5 Hz, we can distinguish the 24 users from each other with over 60% accuracy.

Singular Spectrum Analysis (SSA) [204] is a model-free technique that decomposes time-series into the trend, periodic, and structureless (or noise) components using singular value decomposition. In our case, we decompose $\boldsymbol{X} = \{\boldsymbol{X}^1, \boldsymbol{X}^2, \ldots, \boldsymbol{X}^D\}$ such that the $\boldsymbol{X}^i$ and $\boldsymbol{X}^{i+1}$ are arranged in descending order according to their corresponding singular value and the original time-series can be recovered as: $\boldsymbol{X} = \sum_{i=1}^{D} \boldsymbol{X}^i$. Thus, we test the idea of *incremental reconstruction* by SSA as a base-line transformation method. Figure 4.5 (right) 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.

4.4.2.2 AAE performance

To measure the utility, we train an activity recognition classifier on both the original data and the output of each transformation method: *Resampling*, *SSA*, [86], and the AAE. Then, we use the trained model for inference on the corresponding testing data. Here we use the *Subject* setting, thus the testing data include data of new unseen users. The second row of Table 4.1 (ACT) shows that the average accuracy for activity recognition for both Original 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 original data.

To measure the privacy loss, we assume that an adversary has access to the training dataset and we measure the ability of a pre-trained deep classifier on users' original data in inferring the identity of the users when it receives the transformed data. We train a classifier in the *Trial* setting over original data and then feed it different types of transformed data. The third row of Table 4.1 (ID) 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 has had access to the original data of the users cannot distinguish them after applying the transformation.

Finally, to evaluate the privacy loss and efficiency of the anonymization with an unsupervised mechanism, we implement the k-Nearest Neighbors (k-NN) with Dynamic Time Warping (DTW) [205]. Using DTW, we measure the similarity between the transformed data of a target user k and the original data of each user l, $\mathbf{X}^l$, for all $l \in \{1, \ldots, k, \ldots, N\}$. Then we use this similarity measure to find the k nearest neighbors of user l and check their rank. The last row of Table 4.1 (DTW) shows that it is very difficult to find similarities between the transformed and original data of the users as the performance of the AAE is very similar to the baseline methods and the constraint in Equa-

Table 4.1: Trade-off between utility (activity recognition) and privacy (protecting identity). The second and third rows show the accuracy of the trained classifier which illustrate how data, in each column, is informative about activity and identity, respectively; as a proxy to the mutual information between each time window and the corresponding labels in Equation (4.2.3). The fourth row shows the K-NN rank between 24 users (the lower the better). Key – ACT: activity recognition, *ID*: identity recognition, ACC: accuracy, *F1*: $F1-score$, DTW: Dynamic Time Warping as similarity measure, *SSA*: Singular Spectrum Analysis, AAE: Our method.

Experiment	*Measure*	*Original Data*	*Resampling*		*SSA*		[86]	AAE
		50Hz	*10Hz*	*5Hz*	*1+2*	*1*	*50Hz*	*50Hz*
ACT	mean F1	92.5	91.1	88.0	88.6	87.4	91.5	**92.9**
	variance F1	2.1	0.6	1.8	0.9	0.9	0.9	**0.37**
ID	mean ACC	96.2	31.1	13.5	34.1	16.1	15.9	**7.0**
	mean F1	95.9	25.6	8.9	28.6	12.6	11.2	**1.8**
DTW	mean Rank	0	7.2	9.3	6.8	9.5	10.7	**6.6**
	variance Rank	0	5.7	5.8	5.6	5.4	5.5	**4.7**

tion (4.2.3) maintain the data as similar as possible to the original data. This result shows that the utility-privacy trade-off of AAE is preferable to that of the other methods.

4.4.2.3 Visualization

To gain an understanding of the type of distortions introduced by the AAE, we visually compare sensor time windows before and after transformation.

Figure 4.6 (top) shows the low-dimensional latent representation of original gyroscope data extracted by the bottleneck of the model. The distribution of $\mathbf{Y}$ has useful information to distinguish not only the activities, but also the users (color clusters of the top-right plot). We can notice that just using dimensionality reduction methods cannot ensure anonymization even if we greatly reduce the dimensions. Figure 4.6 (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 4.7 compares original and transformed data of four activities. It is noticeable that the AAE obscures patterns and peaks, but maintains differences among data of different activities. Finally, Figure 4.8 compares the spectrogram of original 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.

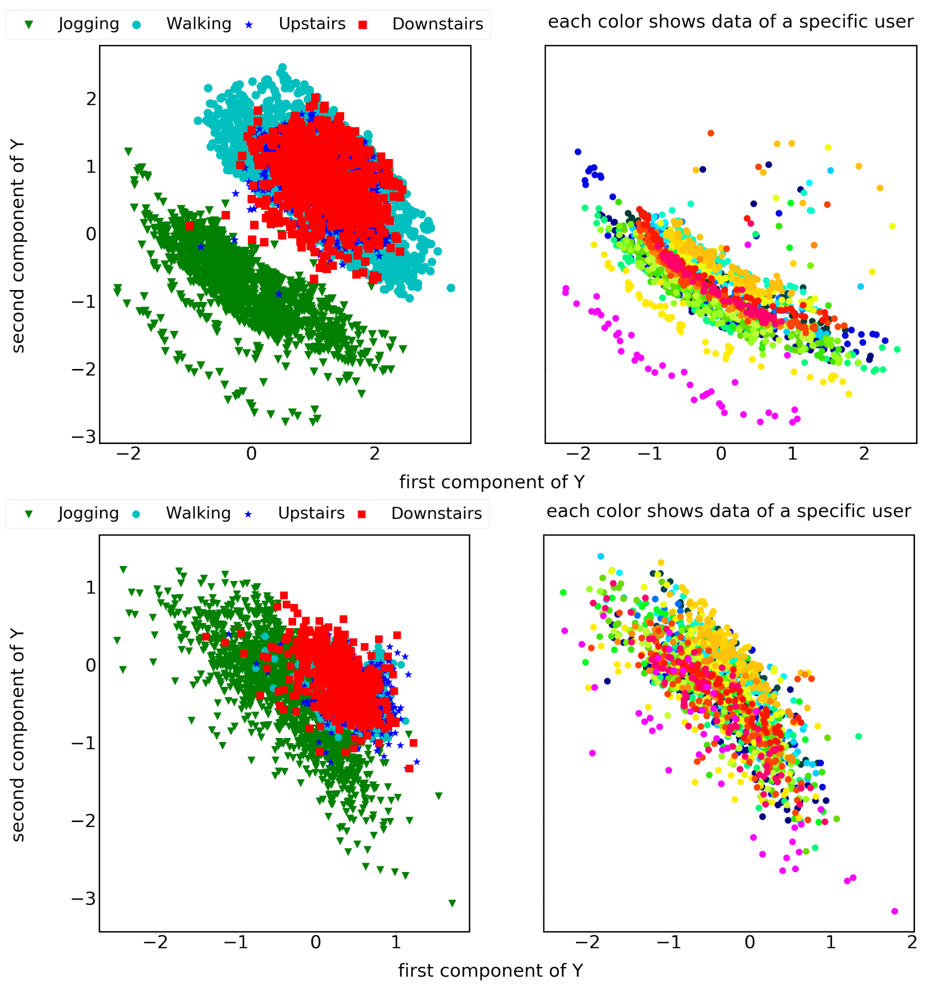


Figure 4.6: A 2D visualization of the lower-dimensional representation of the input data, $\boldsymbol{z}$, in Figure 4.2. (Top row): original data. (Bottom row): data transformed by the AAE: (left) samples of four activities, and (right) *jogging* data for all users.

4.4.3 Protecting Gender Information

Here, we consider the recognition of the user's gender as the sensitive information, whereas the recognition of their activities as the required information. We evaluate a setting where anonymization with the AAE follows replacement using RAE (Chapter 3). An important difference between RAE and AAE is in the nature of sensitive information that they aim to hide. For RAE, each time window, that corresponds to a user's activity, is either sensitive (that must be replaced) or not (that must be released with minimum perturbation), whereas for AAE all time windows include both sensitive information (that must be hidden) and non-sensitive information (that must be kept). The goal is to measure

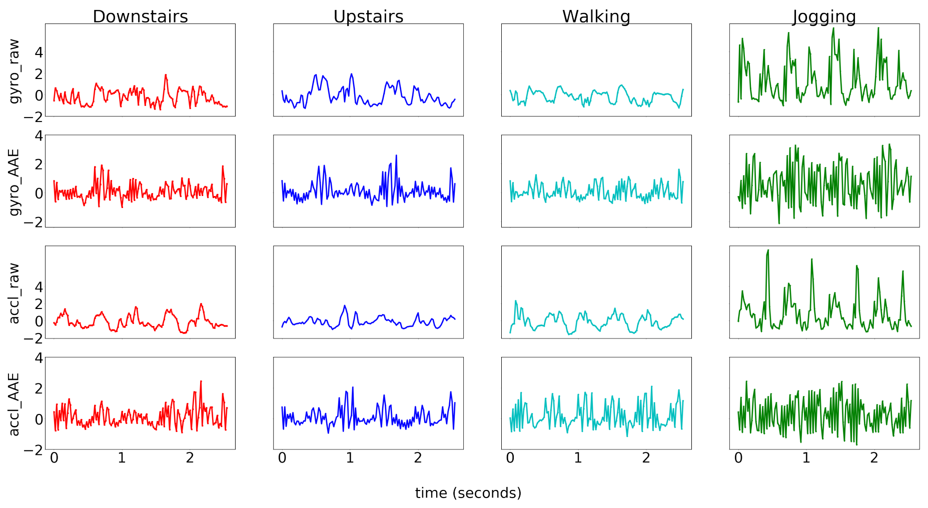


Figure 4.7: Comparison of original (first and third row) and transformed data (second and fourth row) for gyroscope (first two rows) and accelerometer (last two rows) for four activities.

the accuracy of a server in the recognition of the user's sensitive activities and gender when they have access to the original data, versus when they have access to the transformed one (see Figure 4.9).

In MotionSense dataset, we want a server to be unable to infer gender or jogging activity from motion data. Let $\boldsymbol{s}$=\{*jogging*\} be the *sensitive* activity to be replaced with $\boldsymbol{n}$=\{*standing still*\} as the *neutral* activity. We also consider $\boldsymbol{r}$ =\{*walking, stairs-down, stairs-up*\} as the *required* inferences. Let the time-window be 2.56 seconds (W = 128 samples) and H = 2, *i.e.* we consider the magnitude of rotation and acceleration of the device.

First, we train two convolutional neural networks (see Figure 4.3), one as an activity classifier and another as a gender classifier, on the original training dataset. These two classifiers will simulate a potential activity recognition app that may also try to infer the user's gender. We tested multiple settings of hyper-parameter selection and chose the classifiers that achieved the best accuracy on the validation dataset for activity and gender recognition. We compute the accuracy of these models on the test dataset without any modification and report the accuracy in the second column of Table 4.2. Second, RAE is trained to replace the jogging time-windows while keeping the *required* time-windows unmodified in the RAE's output, $\mathbf{X}'$. Third, we use the RAE's output, $\mathbf{X}'$, as the AAE's input and train the AAE to reduce the likelihood of the user's gender being inferred from the ultimate data that is shared with the app, $\mathbf{X}''$. Finally, after training both autoencoders, we feed the testing dataset into the compound model.

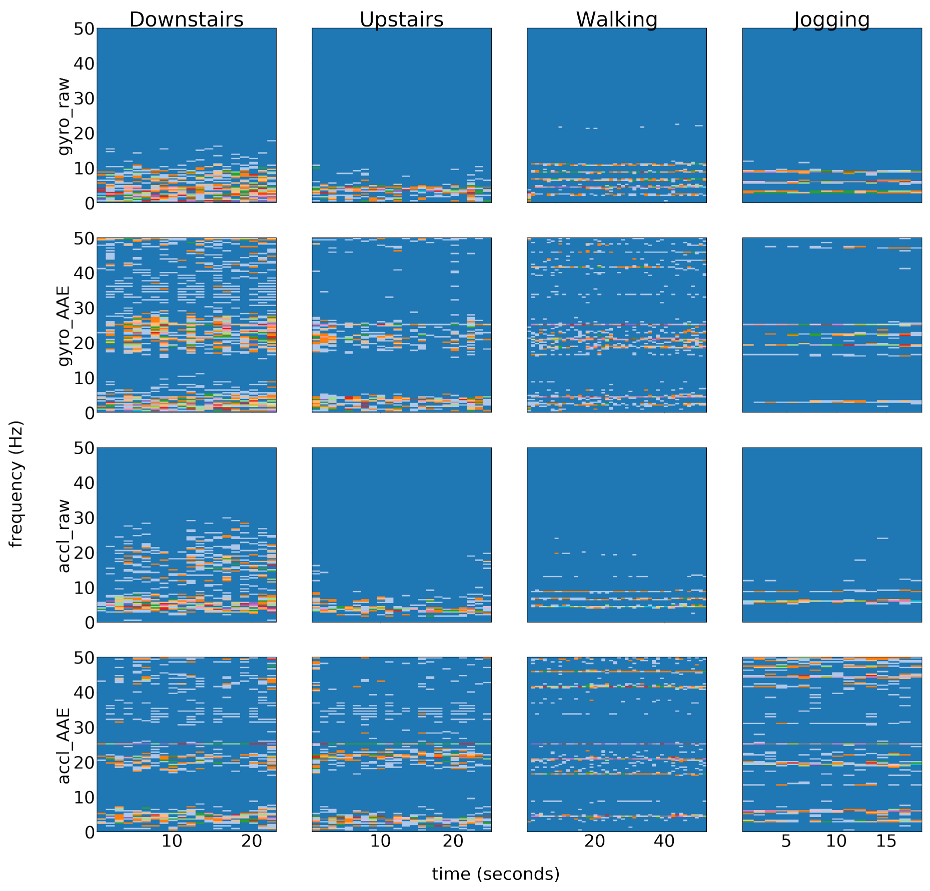


Figure 4.8: Spectrogram of original (first and third row) and transformed data (second and fourth row) for gyroscope (first two rows) and accelerometer (last two rows) for four activities.

Table 4.2: True-positive rate for each activity and gender classification accuracy (%) using a convolutional neural network for each stage of the compound model on MotionSense dataset. "92 as $\boldsymbol{n}$", and so, means 92% of sensitive data misclassified as a neutral data

Inference		$\mathbf{X}$: *Original*	$\mathbf{X}'$: *Replacement*	$\mathbf{X}''$: *Anonymization*	
				$\beta_i = \beta_a = \beta_d$	$\beta_i = \frac{1}{2}\beta_a = \beta_d$
	stairs-down	98.0	93.9	98.5	96.3
$\boldsymbol{r}$	stairs-up	96.4	97.8	92.3	96.3
	walking	99.7	94.8	89.4	94.8
$\boldsymbol{s}$	**jogging**	**99.3**	**1.4 (92 as $\boldsymbol{n}$)**	**0.2 (92 as $\boldsymbol{n}$)**	**0.1 (84 as $\boldsymbol{n}$)**
$\boldsymbol{n}$	standing	99.9	99.9	100	99.9
Gender		**98.9**	**97.1**	**45.0**	**39.0**

Table 4.2 shows the activity and gender classification results at each processing stage. While $\mathbf{X}$ is highly informative for all inferences, after replacement *jogging* intervals are not inferred in $\mathbf{X}'$ and they are classified as *standing*. However, *gender* can still be inferred from $\mathbf{X}'$. Inferring gender from $\mathbf{X}''$ (*i.e.* after

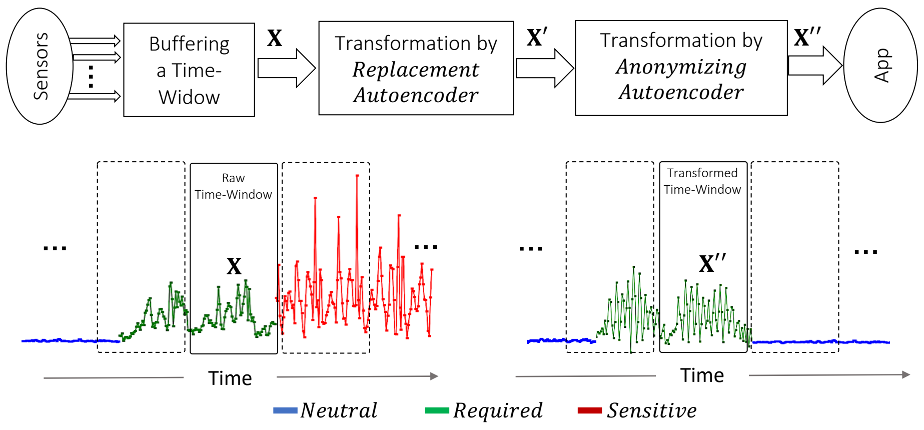


Figure 4.9: (Top) the data flows in the compound framework. At the test-time, first RAE automatically replaces *sensitive* time-windows with non-sensitive *neutral* data, while *required* time-windows are passed with minimal distortions. Then, AAE transforms data to reduce the chance of the user's *gender recognition*. (Bottom) a visual illustration of our transformation mechanism. Depicted signals show accelerometer data transformation for standing, walking, and jogging activities respectively as *neutral*, *required*, and *sensitive* inferences (from the experiment of Table 4.2).

anonymization) reaches the desired level of random guess while the inference of $\boldsymbol{r}$ is maintained close to the original accuracy. Importantly, the proposed framework allows us to give different weights on preserving the activity and hiding gender: the last column of Table 4.2 shows that a better accuracy can be obtained if we increase the risk of leaking more sensitive information. Notice that, the random guess is 58% accurate in this dataset ($\frac{14 \text{ males}}{24 \text{ males and females}}$). Thus, the privacy loss is larger when we have 39% accuracy for gender classification than 45%.

We repeat the same experiment as Table 4.3 on MobiAct [175] dataset by keeping the same architecture for RAE and AAE. In this experiment we consider all kinds of *falls* as *sensitive* activities, assuming that they can be considered as symptoms of some diseases. We also consider being *steady* as a *neutral* activity, which in this dataset it means either sitting or standing. We see results for two different settings for utility-privacy parameters in Table 4.3, that show almost similar results to what we have on the MotionSense dataset in Table 4.2. Accuracy of recognizing the *required* activities are almost the same before and after transformations, while accuracy in detecting falls is dropped from 99.6% to less than 4.5%. Moreover, we can reduce the adversary's accuracy in detecting gender from 97.35% to 66.8%, which is close to the random guess in this dataset that is 74.5% ($\frac{41 \text{ males}}{55 \text{ males and females}}$).

Table 4.3: True-positive rate for each activity and gender classification accuracy (%) using a convolutional neural network for each stage of the compound model on MobiAct [175] dataset.

Inference		$\mathbf{X}$*: Original*	$\mathbf{X}'$*: Replacement*	$\mathbf{X}''$*: Anonymization*	
				$\frac{1}{10}\beta_i = \beta_a = \frac{1}{5}\beta_d$	$\frac{1}{4}\beta_i = \beta_a = \frac{1}{2}\beta_d$
$\boldsymbol{r}$	stair-stepping	98.5	98.4	98.2	98.6
	walking	97.8	96.9	96.7	94.1
	jogging	94.5	93.4	92.1	93.3
	jumping	93.2	93.2	91.4	89.6
$\boldsymbol{s}$	**falling**	**99.6**	**3.6 (96.1 as $\boldsymbol{n}$)**	**3.4 (95.9 as $\boldsymbol{n}$)**	**4.4 (94.9 as $\boldsymbol{n}$)**
$\boldsymbol{n}$	steady	98.6	98.5	95.8	92.7
Gender		**97.3**	**95.5**	**79.9**	**66.7**

4.5 Summary

In this chapter we showed how to train a deep autoencoder, called anonymizing autoencoder (AAE), using a multi-objective loss function for transforming multivariate sensor data such that an activity recognition app cannot discover the user's identity or gender. Experiments conducted on real-world sensor data show that the AAE obscures user-specific motion patterns that enable user re-identification, introducing a small utility loss for activity recognition tasks. We trained AAE in an adversarial setting [86, 89] where we give the transformed data, by AAE, to another neural network model that classifies the sensitive variables. We showed how such an adversarial model can approximate the mutual information between transformed data and sensitive data.

To remove sensitive patterns in data, AAE not only forces the encoder to produce an uninformative latent representation, but also forces the decoder to reconstruct data that cannot be confidently assigned to a specific user in the training set, so the final trained model can generalize to a new unseen user. Moreover, using some constraints on the amount of the allowed data distortion, AAE tries to reconstruct data such that it is useful for an activity recognition app. We showed that AAE achieves a better utility-privacy trade-off compared to approaches that only consider regularizing the latent representation [86]. Unlike other solutions [163, 149, 73, 74] that need a trusted party to collect all the users' data and then run a one-shot privacy-preserving algorithm, the AAE can anonymize data locally and can be shared across users. Thus, the AAE has application in participatory sensing [206], to ensure anonymization for participants who contribute their personal data for health and well-being applications.

Chapter 5

Privacy-Preserving Contextual Bandits

5.1 Overview

In this chapter, we look into the privacy concerns corresponding to user's web-browsing temporal data that is utilized for *personalization* purposes. Personalization aims to improve the quality of the content recommendations (*e.g.* news articles or advertisements) by dynamically adapting to a user's interests and requirements. However, personalization algorithms learn how to maximize a user's positive responses to the proposed content, through time, by collecting potentially sensitive information about the user's interests and their past responses, a fact that raises privacy concerns. For example, as shown in Section 2.1.4, a web browser captures a history of the visited URL by a user in a temporal *histogram* that specifies the user's engagement in different URLs and can be mapped to the user's temporal needs and interests; a fine-grained personal data that can reveal the identity of the user.

Contextual Bandit Algorithms (CBAs) [41, 134] are one of the major tools for such personalized services. A *centralized* CBA, running on a centralized server, analyzes each user's temporal *contextual* data (*e.g.* web-browsing histogram) to recommend them some *content*. Then, based on the user's responses to the recommended content, the CBA can improve its future recommendations. As centralized CBAs collect potentially *sensitive* data, such as the web-browsing history, a privacy-preserving solution should be used to keep and process contextual data locally. Hence, we focus on *local* CBAs that run on the user's local device and perform on-device recommendations. Thus, as sensitive data does not leave the user's device, this approach naturally maintains the user's privacy[1].

The main challenge is that while a local CBA can achieve perfect privacy, this approach is detrimental to personalization, as it fails to incorporate useful information gleaned from other users, limiting its utility in making good recommendations. This is known as the *cold-start* problem, where each local CBA trains its algorithm from scratch by only observing its own user behavior. Hence, the quality of initial recommendations to the user is often too low because the local CBA needs to explore more into different content to learn a good model of its user's interests.

In this chapter, we propose *Privacy-Preserving Bandits* (P2B): a system that updates local CBA agents by collecting feedback from other agents in a differentially-private [60] manner. Therefore, while we keep data and run agents locally, we also allow them to collaborate to solve the cold-start problem. Comparisons of P2B with the other two extreme approaches, non-private CBAs and fully-private (local) CBAs, show competitive performance on both synthetic benchmarks and real-world data.

We evaluate P2B on three types of datasets that (explicitly or implicitly) simulate a recommendation system: synthetic dataset, online advertising dataset, and multi-label classification dataset. Specifically, while providing differential privacy (DP) with a privacy budget $\epsilon \approx 0.693$, P2B only decreases the multi-label classification accuracy by 2.6 and 3.6 percentage point, and keeps the click-through rate (CTR) in a content advertising dataset almost the same as the non-private counterpart. These results suggest P2B is an effective approach to challenges arising in on-device privacy and utility preserving personalization[2].

5.2 Bandit Agents

5.2.1 Learning from Interactions

One of the most natural approaches to learning is to model a learner as an *agent* that learns from its interactions with the *environment*; known as *Reinforcement learning* [207]. Overall, reinforcement learning deals with learning how to map the environment's *state* to an agent's *action* such that the achieved *reward* from the environment is maximized. Naturally, actions not only affect the immediate reward but more or less the subsequent rewards. This is a different approach than *supervised learning*. The goal of supervised learning is to train an agent using a labeled dataset such that it can be generalized well when, at the test time, it is supposed to correctly label an unseen data sample from a similar

[1]In a practical setting, we still need to do extra work to eliminate potential security attacks.

[2]Code and data to reproduce results are publicly available at `https://github.com/mmalekzadeh/privacy-preserving-bandits`

distribution of the training dataset. In reinforcement learning, we do not have access to a labeled dataset that well represents all possible situations that an agent may face in the environment. Therefore, instead of entirely relying on supervision, an agent is supposed to learn from its own interactions.

In learning from interactions in an active environment, the agent is always faced with the overall trade-off between *exploitation* and *exploration*. The former means the desire to take actions that have been awarded better in the past or recent interactions, while the latter means to choose actions that are not well experienced yet, in order to discover other potentially good actions for future exploitation. Exploration is an essential strategy for an active agent, because it helps to experience interactions that are impossible to see if the agent always greedily selects actions that have the best temporal reward. For example, recognizing that a (potentially student) user is interested in the educational content, the agent can always recommend education-related materials during the whole period of the service. However, purely focusing on the exploitation of what the agent knows about the user will prevent the agent to explore what it does not know, especially when the user's interest is dynamically changing through time.

A *policy* (*i.e.* algorithm) specifies when and how the agent decides to exploit or explore at a given round. A policy's input is the current state of the environment, and its output is the action to be taken. Mostly, policies are stochastic such that they output a probability distribution over the set of possible actions. After taking an action, the agent receives a reward from the environment. The agent's temporal reward is usually coming from a stochastic function of the states of the environment and the actions that are taken. The objective of a policy is to maximize the total reward that the agent receives over the long run [207].

5.2.2 Contextual Bandit Algorithms

A *multi-armed bandit problem* is a special case of the reinforcement learning problem in which the environment is *non-associative*: an environment that has only a single state. In a non-associative environment, the reward function is characterized by a set of fixed probability distributions, each corresponding to one of the possible actions (*a.k.a.* arms[3]). For example, in a multi-armed bandit problem with a (fixed but unknown) Bernoulli reward function, for each action a the environment releases a reward of 1 with probability $\mathtt{P}_a$, and otherwise a reward of 0. In this case, the agent explores to find the action with the highest $\mathtt{P}_a$, and then it exploits it for the rest of interactions. If the Bernoulli reward

[3]An analogy to a slot machine that has multiple arms (levers) to choose and pull.

function is not fixed, then the agent needs to perform occasional explorations to keep track of the best action as it changes over time.

A more realistic environment is an *associative* environment, where there is more than one state. In a *contextual bandit problem* the agent seeks to learn a policy that maps each possible state to the action that is probably the best in that state. A *Contextual Bandit Algorithm*, at each round $t \in \{1, 2, \dots, T\}$, selects an action $a_t \in \{1, 2, \dots, A\}$ based on the observed d - dimensional context[4] vector $\boldsymbol{x}_t = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_d]^\intercal$ at that round. In our web-browsing example, a context vector is the (normalized) histogram of the user's engagement in different content. For instance an $\boldsymbol{x}_t = [.4, .2, .1, .3]^\intercal$ can represent the user's engagement in content categories such as [*shopping*, *social media*, *sport*, *education*]. It is important to notice that a context vector $\boldsymbol{x}_t$ is not showing a specific URL visit at round t, but it is showing a cumulative histogram of all the URLs that are visited by the user at round t. Thus, in this example, 0.3 means 30% of the user's URL visits are educational websites.

The agent then obtains the reward $r_t \in \{0, 1\}$ associated with the selected action, without observing the rewards associated with alternative actions. For example, after observing current $\boldsymbol{x}_t$, the agent may recommend a promoted online course, as the action a_t, and receive a positive response, $r_t = 1$.

The main difference between a contextual multi-armed bandit problem and a full reinforcement learning problem is that the chosen action by a CBA affects only the agent's immediate reward, but in a full reinforcement learning problem, it may affect the reward corresponding to the next context as well. A fact that makes it much more difficult to deploy a full reinforcement learning agent in practice.

5.2.3 Upper Confidence Bound Algorithm

As explained, to deal with uncertainties in non-fixed associative environments, an agent needs to explore. A simple algorithm, called *explore-first*, uniformly explores all actions, each for N times, and then picks the empirically best action for exploitation. A better algorithm, called *ϵ-greedy*, in each iteration randomly and uniformly chooses an action to explore with probability ϵ, otherwise, it greedily chooses the current best empirical action with probability $1 - \epsilon$. This helps the agent to spread explorations over time, instead of only at the first N iterations [115]. A problem with ϵ-greedy is that the agent will continue exploring bad actions even when they already have been explored multiple times.

To make explorations more efficient, one idea is to give more chance to actions that are not well explored yet, and the agent has higher uncertainty

[4]In the literature of bandit algorithms, people usually use the term *context*, instead of *state*, where both refer to the same concept.

about their rewards than other actions. This approach is called *upper confidence bound* (UCB) [208] that computes upper bounds on the plausible rewards of each action and consequently selects the action with the highest bound.

For contextual bandits, the agent should also take the current context $\boldsymbol{x}_t$ into account. Linear UCB [41, 209] is a contextual algorithm that computes UCB based on a linear combination of contexts and rewards encountered on previous iterations to propose the appropriate action for the current context. For each action a, Linear UCB keeps a set of trainable parameters $\boldsymbol{W}_a \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b}_a \in \mathbb{R}^d$ which are updated at each iteration. Briefly, at each iteration, the next action is chosen to be:

$$a_t = \underset{a \in \{1,2,\dots,A\}}{\arg\max} \, (\boldsymbol{W}_a^{-1} \boldsymbol{b}_a)^\intercal \boldsymbol{x}_t + \alpha \sqrt{\boldsymbol{x}_t^\intercal \boldsymbol{W}_a^{-1} \boldsymbol{x}_t}$$

where $\alpha \geq 0$ is the parameter that controls exploration-exploitation trade-off, and $\boldsymbol{x}_t$ and $\boldsymbol{b}_a$ are column vectors. Then, based on the user's response to the proposed a_t (*e.g.* to click or not to click on a recommended content), the agent will update parameters $\boldsymbol{W}_a$ and $\boldsymbol{b}_a$ based on the Linear UCB algorithm proposed in [41].

5.3 P2B Methodology

Privacy-preserving data collection has numerous applications. To collect browser settings data, Google has built RAPPOR into the Chrome browser [137]. RAPPOR hashes textual data into Bloom filters using a specific hash function. It then randomizes the Bloom filter, which is a binary vector and uses it as the permanent data to generate an instantaneous randomized response for sharing with the server. RAPPOR can be used to collect some aggregated statistics from all users. For example, having a huge number of users, RAPPOR can estimate the most frequent homepage URLs. However, since the shared data is only useful for estimating aggregated statistics like mean or frequency, the utility of every single shared data for training a model is often too low; the accuracy of the private data becomes unacceptable even with a large number of users.

As an extension to RAPPOR, the Encode-Shuffle-Analyze (ESA) architecture [140, 158] adds two more layers to a local DP algorithm, called shuffler and analyzer, that aims to obscure the identity of the data owner through oblivious shuffling with trusted hardware. ESA implemented in PROCHLO [140, 158] promises to safeguard user's privacy, while preserving the quality of resulting recommendations, through a combination of cryptographic, trusted hardware, and statistical techniques. The Shuffler in the ESA architecture eliminates all metadata attached to the users' reports to eliminate the possibility of linking a data report to a single user during data collection. However, if users do not want to trust any other party, the provided privacy will be the same as RAPPOR.

We explore the question of how the ESA approach can be used in a distributed personalization system to balance the quality of the received recommendations with maintaining the privacy of users. We propose P2B—Privacy-Preserving Bandits—a system where individual agents running locally on users' devices are able to contribute useful feedback to other agents through centralized model updates while providing DP guarantees [68, 60]. To achieve this, P2B combines a CBA agent running locally on a user's device with a privacy-preserving data collection scheme similar to ESA. For the CBA of the local agent, we implement the well-known linear UCB algorithm [41]; which is one of the most popular algorithms for recommendation systems in practice [115]. Moreover, updating the weights of a CBA agent only needs knowledge about the most recent interaction with the user, which makes them more practical for a lightweight local recommendation agent, than other full RL agents [207].

In P2B, every user runs their own CBA agent that works independently of any other agents[5]. At round t the agent observes the current context $\boldsymbol{x}_t$, which represents the user's recent web browsing history. Based on $\boldsymbol{x}_t$, the agent proposes an action a_t, and consequently observes the reward r_t associated with the action (*e.g.* proposing a news article and observing the user's reaction to that article). As the interaction proceeds locally, we refer to agents running on a user's device as *local agents*. The agent may periodically elect to send some information about an observed interaction to a data collection server, and we show how agents encode their data such that the released data to the server satisfies differential privacy with a specific bound. Using the collected data, the server updates a central model that is then propagated back to the users. Figure 5.1 summarizes the overall architecture of the proposed framework. The rest of this section presents P2B's operation and details its various components. Section 5.4 describes how P2B satisfies DP, and Section 5.5 experimentally explains how this approach improves the performance of local agents.

5.3.1 Randomized Data Reporting

Local agents participate in P2B through a randomized participation mechanism. After having some local interactions with the user (*e.g.* after recommending $T \geq 1$ different content), the local agent may randomly construct a payload, containing an encoded instance of interaction data, with probability p. Randomized participation is a crucial step in two ways. First, it raises the difficulty of re-identification attacks by randomizing the timing of recorded interactions.

[5]Note that *user* is an individual and *agent* is an on-device algorithm that serves a user.

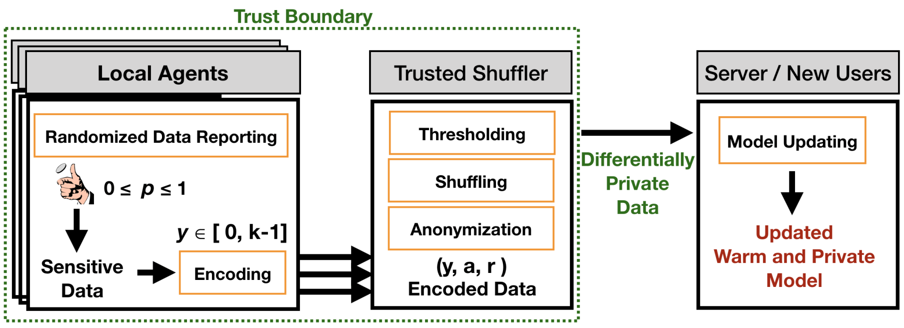


Figure 5.1: System architecture for P2B. A local agent, with probability p, encodes a context vector $\boldsymbol{x}_t$ into the code y and sends it to the shuffler, alongside the recommended action a and observed response r. The trusted shuffler removes all meta-data that may help to deanonymize data, filters data with a frequency lower than the pre-specified threshold l to satisfy the crowd-blending privacy model, shuffles the received data to mitigate potential timing attacks, then sends all data to the server that updates the CBA algorithm to share it with other users and mitigate the cold-start problem.

More importantly, the participation probability p as a source of randomness, has a direct effect on the DP parameters ϵ and δ we will establish in Section 5.4. Briefly, by choosing the appropriate participation probability, one can achieve any level of desired privacy guarantee in P2B.

5.3.2 Encoding

The agent encodes an instance of the context vector, $\boldsymbol{x}$, prior to data transmission. The encoding step acts as a function that maps a d-dimensional context vector $\boldsymbol{x}$ into a code $y \in \{0, 1, \ldots, K-1\}$, where K is the total number of encoded contexts. Here, $\boldsymbol{x}$ denotes the original and highly informative contextual data, for instance, a histogram with d bars that shows the user's engagement in different content types (*e.g.* URLs). On the other hand, y is considered as a cluster code for $\boldsymbol{x}$; a highly restricted contextual data that is aimed to be useful to a CBA for distinguishing different states of the user's behavior, but not useful to distinguish a user in a population.

Let us consider $\boldsymbol{x}$ as a histogram with d bars, where the value of each bar has a fixed precision of q decimal digits and the histogram is normalized that means its values sum to 1. This combination of normalization and finite precision has two important characteristics. First, it allows us to precisely enumerate all possible contexts according to a classic theorem of counting problem in combinatorics, that is proved by a technique called as *stars and bars* [210]. The theorem says that for any pair of positive integers s and d, the number of d-tuples of non-negative integers whose sum is s is equal to the number of multisets of

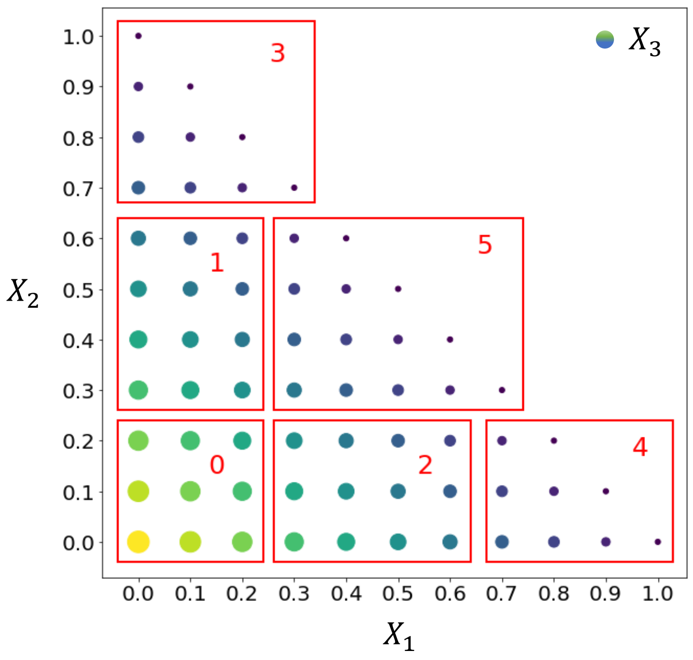


Figure 5.2: Normalized vector space of $\boldsymbol{x} = (x_1, x_2, x_3)$ with precision $q = 1$ that has the cardinality $n = 66$. Context vectors are normalized such that $x_1 + x_2 + x_3 = 1$. The size of the circles shows the value of x_3. Rectangles show a potential encoding of the vector space with $K = 6$ clusters. Here, the minimum number of elements in a cluster is $l = 9$ which is an important parameter for the crowd-blending privacy model.

cardinality $d - 1$ taken from a set of size $s + 1$ that is

$$\binom{s + d - 1}{d - 1}.$$

In our setting, using a finite precision of q decimal digits we have $s = 10^q$ and the cardinality, n, of the set of possible normalized context vectors is

$$n = \binom{10^q + d - 1}{d - 1}. \tag{5.3.1}$$

Secondly, the samples of such normalized context vectors are distributed uniformly, in a grid shape, in the d-dimensional space. For instance, in Figure 5.2 we see an example of encoding a 3-dimensional normalized vector space that is encoded in $K = 6$ different codes. In this example, the number of 3-tuples of non-negative x_i whose sum is 1 (*e.g.* $[0.3, 0.5, 0.2]$) is $n = 66$.

Given that the CBA agent tends to propose similar actions for similar contexts, neighboring context vectors can be encoded into the same context code y. While this approach may appear to be limiting, as it reduces the granularity of the data, we will show in Section 5.5 that the encoded values are still useful for establishing a desired utility-privacy trade-off. The encoding algorithm can be chosen depending on application requirements, but we limited our experimental evaluation to the simple K-means clustering [211]. After electing to participate and encoding an instance of a user interaction, the agent transmits the data in the form of the tuple (y_t, a_t, r_t) to the *shuffler*.

5.3.3 Shuffler

The trusted shuffler is a critical part of every ESA architecture [140], and in P2B it is necessary for ensuring anonymization and DP guarantee. Following the same PROCHLO implementation [140], the shuffler operates in a secure enclave on trusted hardware and performs three tasks:

1. **Anonymization**: eliminating all the received meta-data (*e.g.* IP address) originating from local agents.
2. **Shuffling**: gathering tuples received from several local agents into batches and shuffling their order. This helps to prevent side-channel attacks (*e.g.* timing attack).
3. **Thresholding**: removing tuples that their encoded context vector frequency in the batch is less than a defined *threshold*. This threshold plays an important role in the achieved DP bound.

After performing these three operations, the shuffler sends the refined batch to the *server* for updating the model. Upon receiving the new batch of training data, the server updates the global model and distributes it to local agents that request it. More importantly, this global model is used as a warm-start by new local agents that are joining the system to help in preventing the cold-start problem.

5.4 Privacy Analysis

This section analyzes how P2B ensures DP through a combination of pre-sampling and crowd-blending (see Section 2.2.2). Here, we assume that users wish to share their *sensitive* context vector $\boldsymbol{x}$ with the server without revealing their identity (*i.e.* who is the owner of $\boldsymbol{x}$), and we discuss how randomized pre-sampling, encoding $\boldsymbol{x}$ into y, and shuffler's operations make P2B a privacy-preserving algorithm. The user's privacy is defined against a curious server that collects the user's context vectors and wants to recognize whether a target user is contributed to the dataset or not. Thus, if we guarantee such DP for our users against the server, then by post-processing property of DP, the same privacy guarantee will hold against all other parties who will get access to the collected data or ML models that are trained on that data.

As context vectors are multi-dimensional real-valued vectors, we assume a context vector in its original form can be uniquely assigned to a specific user. Here, we assume no prior on the possible values for a context vector, meaning context vectors are coming from a uniform distribution over the underlying vector space. This may seem to be an unrealistic assumption, as for example in the real world, not all URLs have the same chance to be visited. However, from the privacy point of view, this uniform assumption indicates a vector space of the maximum entropy which is the worst possible vector space to be encoded into a lower-dimensional space with an entropy depending on the chosen K. With these assumptions, P2B can resist strong adversaries with any kind of prior knowledge or side information, because P2B provides a DP guarantee and, therefore, inherits all the advantages of DP.

For the n different context vectors in Equation (5.3.1), the optimal encoder (see Section 5.3.2) encodes every n/K contexts into one of the possible K codes. Consequently, when a total number of U users participate in P2B to send a tuple to the server, the optimal encoder satisfies crowd-blending privacy with $l = U/K$. In the case of a suboptimal encoder, we consider l as the size of the smallest cluster in the vector space. Furthermore, situations where the number of users is small, leading to a small l, can be addressed by adjusting the shufflers *threshold* to reach the desired l. Essentially, l can always be matched to the shuffler's threshold.

Each user randomly participates in data sharing with probability p (see Section 5.3.1), and then encoding the pre-sampled data tuple. Following [68], the combination of (i) pre-sampling with probability p and (ii) $(l, \bar{\epsilon})-$crowd-blending, leads to a differentially private mechanism with

$$\epsilon = \ln\Big(p \cdot (\frac{2-p}{1-p} \cdot e^{\bar{\epsilon}}) + (1-p)\Big) \tag{5.4.1}$$

and

$$\delta = e^{-\Omega(l\cdot(1-p)^2)}. \tag{5.4.2}$$

Here Ω is a constant that can be calculated based on the analysis provided by [68]. For better understanding of the role of the parameters ϵ and δ, please visit Section 2.2.1 where we elaborate the DP model.

Our encoding scheme provides an $\bar{\epsilon} = 0$ for crowd-blending, as the encoded values for all the members of a crowd is exactly the same. As a consequence, the ϵ parameter of the DP of the entire data sharing mechanism depends entirely

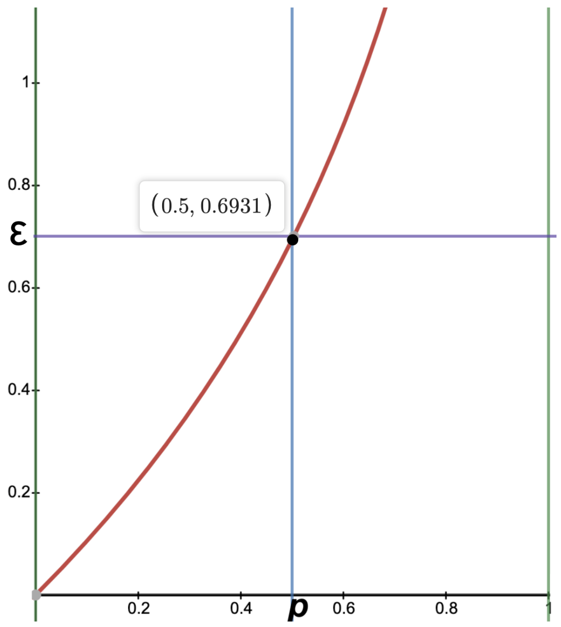


Figure 5.3: ϵ as a result of the probability of participating in the scheme p.

on the probability p of participation as (Figure 5.3):

$$\epsilon = \ln\Big(p \cdot (\frac{2-p}{1-p}) + (1-p)\Big).$$

For example, by trading half of the potential data ($p = 0.5$) P2B achieves an $\epsilon \approx$ 0.693 that is a strong privacy guarantee. On the other hand, δ depends on both p and l. To understand the effect of δ, Dwork *et al.* [60] prove that an (ϵ, δ)-differentially private mechanism ensures that for all neighboring datasets, the absolute value of the privacy loss will be bounded by ϵ with probability at least $1 - \delta$. Therefore, by linearly increasing the crowd-blending parameter l, we can exponentially reduce the δ parameter.

Our illustrations here provide concrete settings and parameters that show how P2B satisfies the requirements of the DP model proposed in [68]. We refer the interested reader to the rigorous analysis and proofs provided in the original paper [68].

It is worth mentioning that a crowd-blending privacy guarantee is not enough per se, and we need a DP guarantee if we need to assure plausible deniability for the users such that it is always possible for the users of P2B to deny participation in the process. For example, in the crowd-blending model, a user can still be identified to certainly be one of l people who their web-browsing history is clustered into a sensitive category (*e.g.* a category showing users who visit some sensitive URLs). However, in the DP model, this identification cannot be done with certainty, as there is always a chance for all users to not be sampled in the first place. Hence, plausible deniability is provided to all users, with a specific statistical bound defined by DP.

5.5 Experimental Evaluation

This section explores P2B's privacy-utility trade-offs compared to (i) non-private and (ii) completely private approaches. The experimental evaluation assumes the standard bandit settings, where the local agent learns a policy based on the contextual Linear Upper Confidence Bound algorithm (LinUCB) [209, 41]. For simplicity, throughout the experiments, the probability p of randomized transmission by a local agent was set to $p = 0.5$, the rounding parameter q of the encoder was set to $q = 1$, and the parameter α for LinUCB was set to $\alpha = 1$, meaning that the local agent is equally likely to propose an exploration or exploitation action. The experiments compare the performance of the following three settings:

Cold. The local agent learns a policy without any communication to the server at any point. As there is no communication, this provides full privacy, but each agent has to learn a policy from a cold-start.

Warm and Non-Private. In this setting, local agents communicate the observed context to the server in its original form. Thus, other agents are able to initialize their policy with a model received from the server and start adapting it to their local environment. This is called warm and non-private start, and represents the other end of the privacy/utility spectrum with no privacy guarantee afforded to the users.

Warm and Private. In this setting, local agents communicate with the server using P2B. Once more, other agents initialize their internal policy with an updated model received from the server and start to adapt it to their local environment. We term this a warm and private start, and the provided privacy guarantees function according to the analysis in Section 5.4.

These approaches are evaluated on synthetic benchmarks, two multi-label classification datasets, and an online advertising dataset.

5.5.1 Synthetic Preference Benchmark

These benchmarks consider the setting where there is a stochastic function $\mathcal{F}$ that relates context vectors with the probability of a proposed action receiving a reward. Specifically, $\mathcal{F}$ is the scaled softmax output of a matrix-vector product of the user contexts with a randomly generated weight matrix $\boldsymbol{W}$ and bias vector

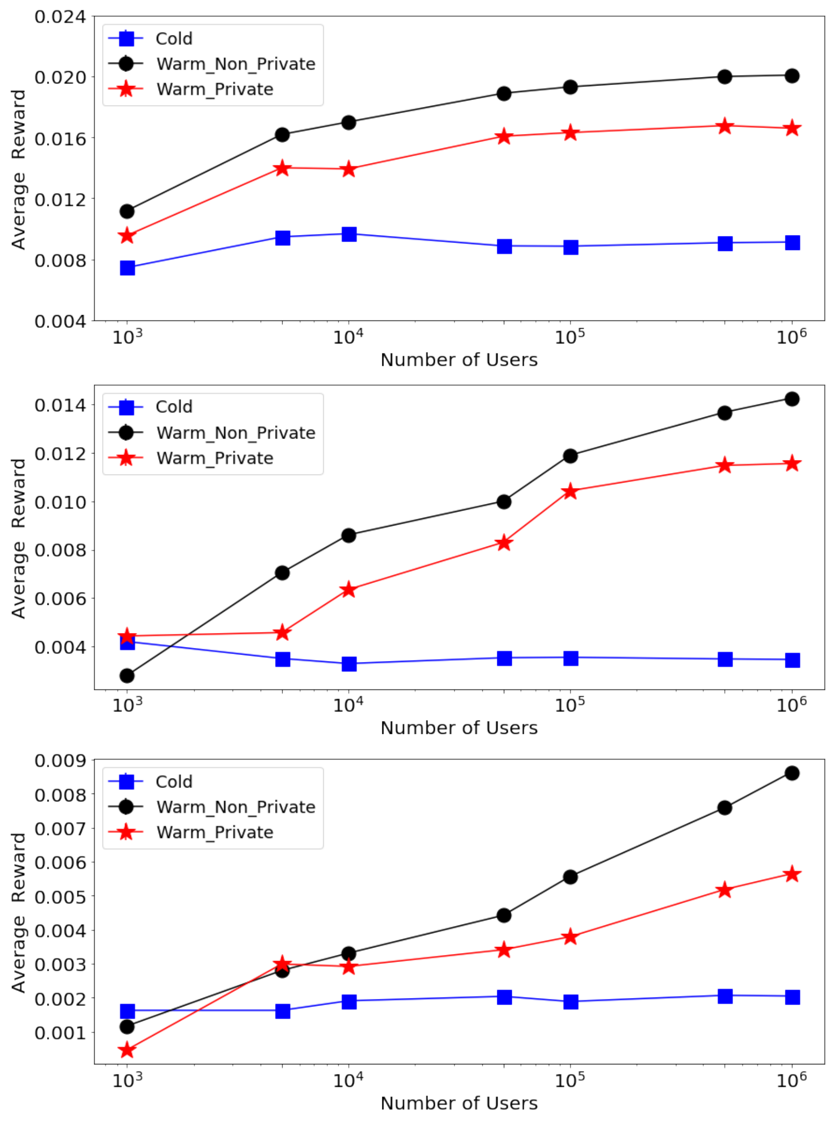


Figure 5.4: Synthetic benchmarks: (Top) $\texttt{size}(\mathbb{A}) = 10$, (Middle) $\texttt{size}(\mathbb{A}) = 20$ (Bottom) $\texttt{size}(\mathbb{A}) = 50$. For all: $d = 10$ and $t = 10$. The expected reward in this setting has a strong dependence on the number of actions as agents will spend considerable time exploring alternative actions.

$\boldsymbol{b}$:

$$\mathcal{F}(\boldsymbol{x}) = \mathsf{softmax}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \in [0,1]^{\texttt{size}(\mathbb{A})}. \tag{5.5.1}$$

For the specific implementation in this section, we use the default weight initializer of TensorFlow [202] to set $\boldsymbol{W}$ and $\boldsymbol{b}$, and then keep them fixed throughout the experiment. In the LinUCB algorithm, we set the mean reward $\bar{r}_a$ for every action a as

$$\bar{r}_a = \beta \mathcal{F}_a(\boldsymbol{x}) + \mathbf{z},$$

where $\mathcal{F}_a$ is the a-th component of the softmax output in Equation 5.5.1, β is a scaling factor with $0 \leq \beta \leq 1$, and $\mathbf{z}$ is random Gaussian noise $\mathbf{z} \sim \mathcal{N}(0, \sigma^2)$. Hence, to create an instance of a synthetic interaction between user and agent, we first randomly and uniformly generate a context vector $\boldsymbol{x}$. Then having $\boldsymbol{x}$ and using $\mathcal{F}$, for each possible action a we calculate the probable reward r. For round t, this process results in a synthetic data tuple $(\boldsymbol{x}_t, a_t, r_t)$.

For all synthetic benchmarks, the parameters are set as follows. The scaling factor for the preference function was fixed to $\beta = 0.1$ and the variance of the Gaussian noise $\sigma^2 = 0.01$. In terms of local agent settings, the agents use $K = 2^{10}$ codes for encoding purposes and observe $t = 10$ local interactions before randomly transmitting an instance with probability $p = 0.5$. We varied the number of dimensions d of the context vector in the range of 6 to 20, and the number of actions in the range of 10 to 50. We observe the average reward in each setting as the user population U grows from 10^3 to 10^6.

Our results in Figure 5.4 indicate that for a small number of interactions the cold-start local model fails to learn a useful policy. In contrast, the warm models substantially improve as more user data becomes available. Utilizing prior interaction data in this setting, more than doubles the effectiveness of the learned policies, even for relatively small user populations. Overall, the non-private agents have a performance advantage, with the private version trailing.

Figure 5.5 illustrates how the dimensionality of the context vector affects the agent's expected reward. By increasing d from 6 to 20 the average reward decreases as agents spend more time exploring the larger context space. P2B remains competitive with its non-private counterparts, and on occasion outperforms them, especially for low-dimensional context settings. The number of actions has a similar effect to the dimensionality of the context, as agents have to spend more time exploring suboptimal actions. Once more, the results of Figure 5.4 indicate that this is the case experimentally.

The number of users is pivotal in the trade-off between privacy and utility. For simplicity, and in order to avoid unnecessary computation, experiments fix $p = 0.5$. However, the expected reward for other p values can be simply deduced from the current experiments. For example, if p changes from 0.5 to 0.25 (meaning ϵ is changed from 0.69 to 0.28) P2B would require $100K$ users rather than $50K$ to achieve the same performance with the new privacy guarantee. In simple terms, to achieve smaller ϵ, a smaller p is required, which consequently means that a proportionally increased number of users is necessary to maintain a similar performance.

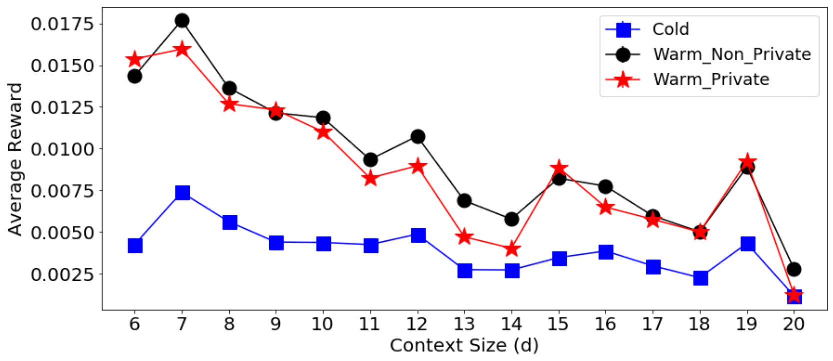


Figure 5.5: Synthetic benchmarks: $U = 20000$, $\texttt{size}(\mathbb{A}) = 20$, $t = 20$, and $d = \{6, 7, \ldots, 20\}$. As the dimensionality of the context increases the average reward for this settings is reduced as agents spend more time trying to explore their environment.

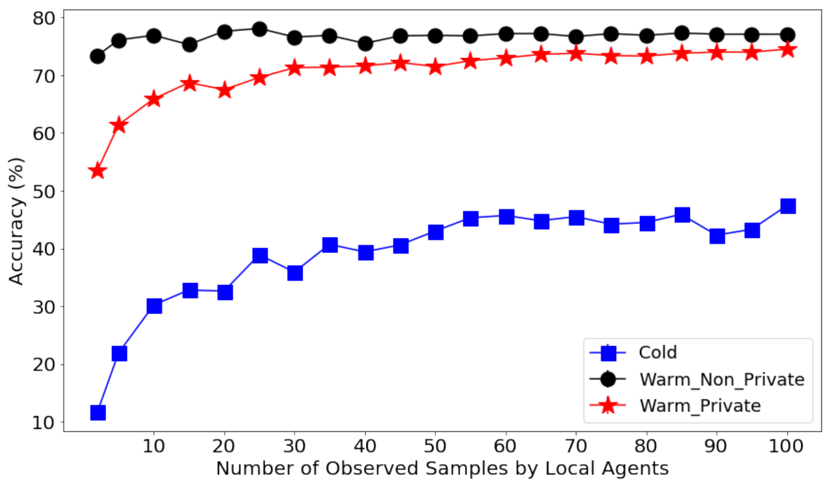
5.5.2 Multi-Label Classification

These experiments examine the performance of the three settings on multi-label classification with bandit feedback. Although a multi-label classification task is not an explicit recommendation task, there are some similarities between these two tasks. In both tasks (classification/recommendation): (i) the prediction is based on the given sample/context (ii) for each sample/context, there usually are more than one correct labels/items to recognize/propose, and (iii) the set of possible labels/items is large enough to not make the problem trivial. Therefore, as we do not have access to many public real-world datasets for the recommendation, it is proposed to use multi-label classification datasets for better evaluation of bandit algorithms [212].

We consider two datasets, namely (1) MediaMill [213], a video classification dataset including 43,907 instances, 120 extracted features from each video, and 101 possible categories, and (2) a TextMining dataset [214] including 28,596 instances, 500 extracted features from each text, and 22 possible categories.

P2B's performance in this setting was evaluated according to the following setting. We consider a fixed number of local agents. Each agent has access to, and is able to interact with, a small fraction of the dataset. In particular, every agent has access to up to 100 samples, which were randomly selected without replacement from the entire dataset. 70% of agents participate in P2B and we test the accuracy of the resulting models with the remaining 30%. For both datasets, the local agents use $K = 2^5$ codes for encoding purposes.

This setting is particularly interesting as it allows us to study how the predictive performance of the system changes when the local agents interact more with the user. In terms of relative performance between the different approaches,



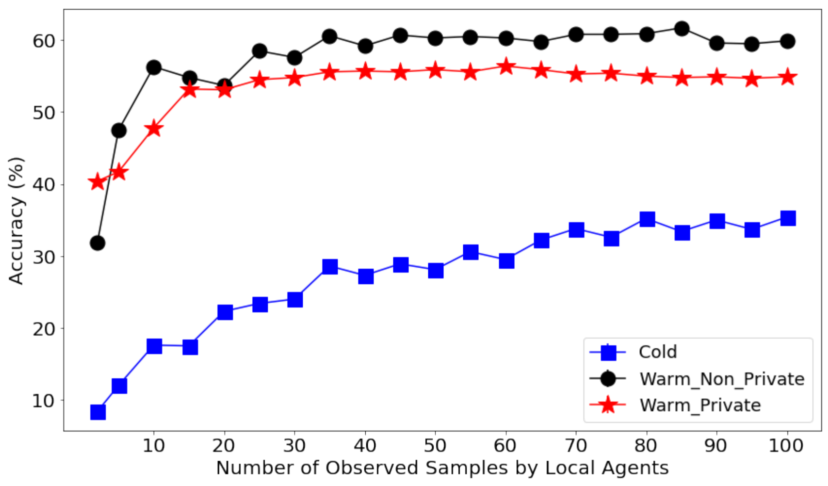
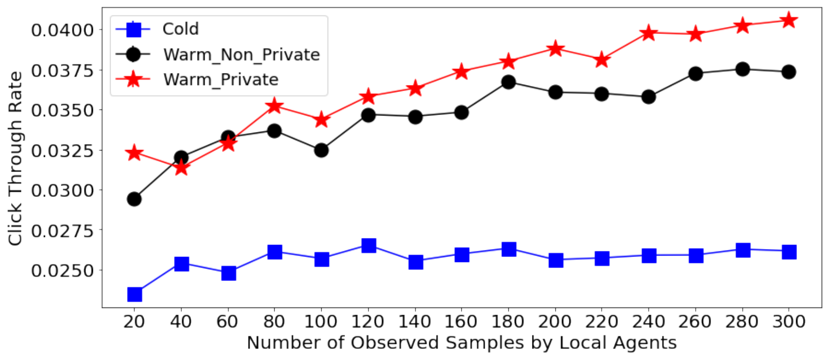


Figure 5.6: Accuracy in multi-label datasets: (Top) Media-Mill with $d = 20$ and $\texttt{size}(\mathbb{A}) = 40$ (Bottom) Text-Mining with $d = 20$ and $\texttt{size}(\mathbb{A}) = 20$. As local agents observe more interactions they obtain better accuracy. This has a multiplicative effect in the distributed settings where agents reach the plateau much faster.

the results in Figure 5.6 repeat the findings of the synthetic benchmarks. The non-private warm version is better than the private warm version, which is still better than the cold version that utilizes only local feedback.

However, we can also observe that the cold version given enough interactions produces increasingly improving results. The centralized update mechanism tends to have a multiplicative effect, especially when there is little local interaction data before reaching a plateau.



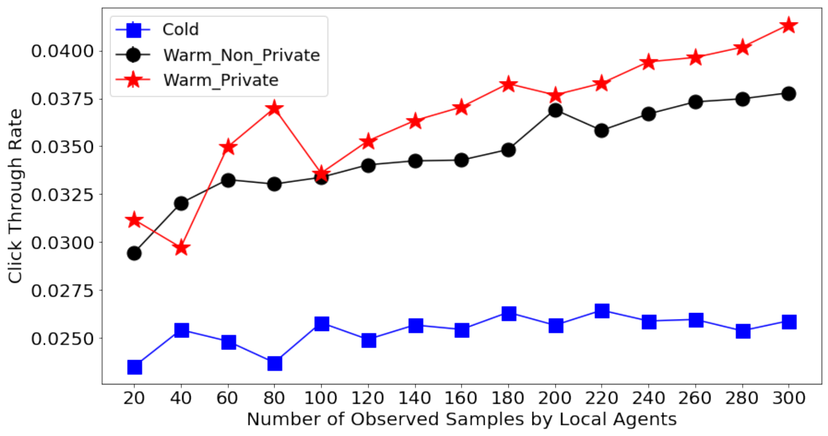


Figure 5.7: Criteo results. $d = 10$, $\texttt{size}(\mathbb{A}) = 40$, (Top) $K = 2^5$, and (Bottom) $K = 2^7$. The private and non-private agents obtain similar performances for low numbers of local interactions. As the number of local interactions increases the private agents perform better than their non-private counterparts.

5.5.3 Online Advertising

Here, we consider an advertisement recommendation scenario where the action is to recommend an ad from one of the existing categories. We use the Criteo dataset from a Kaggle contest[6]. It consists of a portion of Criteo's traffic over a period of 7 days, including 13 numerical features and 26 categorical features. For each record in the dataset, there is a label that shows whether the user has clicked on the recommended ad or not. For anonymization purposes, the feature's semantics are unknown and the data publisher has hashed the values of categorical features into 32 bits.

As the exact semantics of the features were not disclosed, we assume that numerical features represent the user's context and categorical features correspond to the type of the proposed product. For each sample in the dataset, we hash the values of the 26 categorical features into an integer value. The

[6]`https://labs.criteo.com/category/dataset`

resulting integer value is then used as a possible product category to recommend. The hashing procedure operates as follows: First, the 26 categorical values are reduced into a single hashed value using feature hashing [215]. After hashing, the 40 most frequent hash codes are selected. These are converted into an integer value in the range between 1 to 40, based on their frequency (label 1 shows the most frequent code and so on). Finally, for evaluation, we only use data samples having one of these 40 values as the product label and ignore the remaining data.

During the evaluation, the local agent observes the values of the numerical features as the context vector, and in response takes one of the $\texttt{size}(\mathbb{A}) = 40$ possible actions. The agent obtains a reward of 1 if the proposed action matches the logged action in the dataset. The remaining experimental setup for the Criteo dataset is similar to the one used in the multi-label datasets. Specifically, we present experimental results (Figure 5.7) for d=10 and total number of actions $\texttt{size}(\mathbb{A}) = 40$. We compare the results for two values of encoding parameter $K = 2^5$ and $K = 2^7$. The sampling probability p remains 0.5, and the shuffling threshold remains 10. This experimental setting has $U = 3000$ agents, and each of the agents accumulates 300 interactions.

The results in this setting are quite surprising as private agents attain a better click-through rate than their non-private counterparts. There is a recent work [216] showing that privacy-preserving training of machine learning models can aid generalization as well as protect privacy. There are some factors we believe help explain these experimental results.

Private agents use the encoded value as the context. As a result, the number of possible contexts is much smaller compared to using the original d-dimensional context vector. As contextual bandits need to balance exploration with exploitation, especially in the early stages, a bandit with a smaller context size can quickly reach better results. Furthermore, P2B clusters similar context vectors into the same categories and this also helps a private bandit to better act in similar situations. This is an effect that also can be seen for the higher dimensional contexts of the synthetic benchmarks in Figure 5.5.

Once more, the number of users plays a critical role in establishing an acceptable trade-off between utility and privacy. The parameter δ in Equation 5.4.2 can be interpreted as the probability that the worst-case happens in differentially private data sharing. Using a larger K in a system with a small number of users U, limits the crowd blending parameter l and consequently leads to undesired potential values for δ. Unfortunately, the number of users for real-world datasets used in this paper is limited, thus posing constraints on the range of parameters in the experimental evaluation.

5.6 Summary

Taking advantage of clustering algorithms, such as K-means, we proposed a method for efficiently encoding feedback instances of a contextual bandit algorithm on the user's device. We evaluated P2B on a synthetic benchmark, multi-label classification, and online advertising data. We showed experimentally that standalone agents trained on an individual's data require a substantial amount of time to learn a useful recommendation policy. The results experimentally show that P2B remains competitive in terms of predictive utility with approaches that provide no privacy protections. At the same time, as expected, it substantially outperforms on-device cold-start models that do not share data. The experiments show that sharing data between agents can significantly reduce the number of required local interactions in order to reach a useful local policy. We examined the overall trade-offs between recommendation quality and privacy loss by performing a DP analysis of the P2B according to the crowd-blending privacy model [68]. We showed that P2B results in a small ϵ value for DP, which can be directly quantified from the probability of an agent participating in the data collection mechanism. Such bound on the privacy loss, provided by the guaranteed DP, shows that P2B can promise the users strong privacy protection. In a series of experiments, P2B shows considerable improvement for increasing numbers of users and local interactions. With regards to P2B's privacy, experiments show the clustering-based encoding scheme is effective in encoding interactions. In addition, all the experiments relied on a sampling probability $p = 0.5$. This results in a very competitive privacy budget of $\epsilon \approx 0.693$.

Chapter 6

Conclusions

In this chapter, we first provide a summary of this thesis's achievements, then we discuss the research questions that are still open and the corresponding future directions that, we believe, are worth exploring.

6.1 Summary of Achievements

The focus of this thesis was on proposing machine learning algorithms that can help the users of cloud-assisted applications to have more control and protection over the data they share and the amount of information which can be discovered from their data.

First, we proposed two privacy-preserving algorithms for transforming motion-sensor data generated by mobile and wearable devices. (i) We proposed *Replacement AutoEncoder* (RAE) to prevent the inference of the user's sensitive activities without decreasing the accuracy of the desired app in recognizing non-sensitive activities. (ii) We proposed *Anonymization AutoEncoders* (AAE) to avoid the inference of the user's sensitive attributes, or enable the re-identification of the user, from the patterns observed in their motion sensor data. The difference between RAE and AAE is in the nature of sensitive information that they aim to hide. In RAE, each time window corresponds to a user's activity and that activity is either sensitive (that must be replaced) or non-sensitive (that must be released with minimum perturbation), whereas for AAE all time window include both sensitive information (that must be hidden) and non-sensitive information (that must be kept). Despite this difference, we showed how to cascade RAE and AAE into a compound architecture to build a unified framework that can protect both sensitive activities and sensitive attributes, such as gender, that a user may want not to reveal. We evaluated the efficiency of the proposed algorithms on several datasets for human activity recognition, and showed that these algorithms can generalize well for establishing a meaningful privacy-utility trade-off on data of unseen users. Alongside these algorithms, we also collected and published a HAR dataset of motion sensors which is not only used in this thesis, but also by many other researchers.

Second, we proposed *Privacy-Preserving Bandits* (P2B) for efficiently encoding the user's web-browsing data while maintaining its usefulness for collaboratively training personalized content recommendation apps. We showed how P2B can provide a formal differential privacy guarantee while improving the quality of recommendations. We evaluated P2B on several datasets and showed how P2B remains competitive with a non-private counterpart as well as substantially outperforming the non-collaborative fully-private counterpart. Our results suggest P2B is an effective approach to challenges arising in on-device privacy and utility preserving personalization.

6.2 Future Directions

Related to the proposed algorithms in this thesis, there are some directions for future research that we discuss in the following.

1. Collaborative Model Training. We have assumed the existence of a publicly available dataset to train the RAE and the AAE through a trusted mediator. When such a public dataset is not available or users do not trust a mediator, one option is to use privacy-preserving collaborative model training without collecting personal data [143], for example through federated learning [144]. In cases where there is a limited set of the labeled data and a large set of unlabeled data, one option might be to use semi-supervised training using GANs [217], where the unlabeled dataset is used to help the model learning more discerning features for the target task.

2. Diversity and Robustness in Data Transformations. We have seen that the RAE tends more to transform many sensitive activities into one of the possible neutral activities. Such a lack of diversity in replacement might give an adversary some advantages to distinguish whether the received data is a replacement of a sensitive time window or not. Thus, one can investigate an improvement to the RAE that transforms each sensitive activity into a neutral activity that is more probable to happen at that specific time. It should be noted that for evaluation of such improvement one needs to get access to, or collect, a more detailed HAR dataset including time-stamped, consecutive activities data of several users. Deriving a more rigorous statistical analysis for the amount of privacy achieved by the RAE or AAE is an important task. Recent advances in using neural networks and variational bounds [152] for measuring the mutual information between high-dimensional data can be used for better quantifying the amount of preserved/removed information or to introduce new privacy and utility metrics.

3. Cost and Complexity Analysis. Measuring the cost of running the proposed algorithms on the user devices, to see what are the complexities that we should tolerate to locally protect the user's privacy, is an important direction. One can design and evaluate the interplay between the operating system of the user's device, the data transformation algorithms such as RAE, AAE, DANA, or P2B, and the applications that are running on the device and request access to data. Specifically, one can focus on developing algorithms that let users dynamically define their personal privacy policies or resources that the apps are allowed to have access to.

4. A Privacy Model without Losing Data. With the current privacy model of P2B, we need to discard a considerable fraction of the data (*e.g.* half of the data) to ensure an acceptable privacy guarantee. Although the P2B still performs well, there is still some room for working on a privacy model that can take advantage of the whole available data. For example, a differential privacy model using randomized response, instead of randomized pre-sampling, can help. However, as the randomized response needs to add noise to every sample, this approach may scarify more utility than the current approach of P2B.

5. Lower/Upper Bounds for P2B. We have seen that P2B traces a similar trend to its non-private counterpart in several experiments and showed that P2B is competitive and even sometimes outperforms the non-private model. One can also study the behavior of more encoding approaches as well as their interplay with alternative contextual bandit algorithms. Hence, working on theoretical lower- and upper-bounds for the accuracy of P2B to provide more concrete reasons on the behavior of the model is an interesting future direction.

6. Sensor Data Collection. In most of the available HAR datasets, the activities that are categorized into *sensitive*, *required*, and *neutral* are independent of each other and at each time-window, only one of them is happening. However, in a real-world situation there might be correlations among different activities, some activities may happen at the same time, or there might be activities that are more probable to happen consecutively. Moreover, correlations among consecutive time-windows of a specific activity may incrementally reveal information that facilitates the re-identification of the user. To assess these situations for better evaluation of RAE and AAE, we would need access to multi-labeled data collected over a much longer time period as well as a large number of demographically different users. From another point of view, the current public datasets of mobile and wearable sensor data do not simultaneously satisfy the requirements of abundance and variety of activities and users. With larger datasets, one can increase the learning capacity of the proposed deep neural architecture by adding more layers to the neural network or investigate various DNN architectures for RAE or AAE. Collecting and labeling datasets that cover such scenarios can facilitate answering many research questions in this area.

Appendices

Appendix A

Dimension-Adaptive Neural Architecture

A.1 Overview

Motion sensors embedded in wearable and mobile devices allow for dynamic selection of sensor streams and sampling rates, enabling useful applications, *e.g.* for power management or control of data sharing. While deep neural networks (DNNs) achieve competitive accuracy in sensor data classification, current DNN architectures [101, 102, 103, 117, 116, 182, 119, 183, 118] only process data coming from a fixed set of sensors with a fixed sampling rate, and changes in the dimensions of their inputs cause considerable accuracy loss, unnecessary computations, or failure in operation. Thus, current DNNs cannot reliably handle dynamic conditions at inference time (*e.g.* when the sampling rate changes or some sensors are dropped or deselected), which are important for energy preservation [75, 76, 77], privacy protection [4, 73, 74], and fault tolerance [78, 79]. Moreover, DNNs are often trained on datasets collected with specific devices, but may be used in a wider set of devices, with different combinations of sensors and sampling rates [218].

To address sampling rate and sensor selection in a unified framework, we introduce Dimension-Adaptive Neural Architecture (DANA) that is robust to variable sampling rates and sensor selection, and also works on any combination of sensors that were considered at training time. Specifically, we introduce a dimension-adaptive pooling (DAP) layer that captures temporal correlations between consecutive samples and dynamically adapts to all feasible data dimensions. To enable DNNs that use DAP to generalize at inference time over the set of feasible dimensions, we propose a dimension-adaptive training (DAT) procedure, which incorporates dimension randomization and optimization with accumulated gradients. In each forward pass, DAT re-samples a batch of time windows to a new rate and may also cope with removed streams from some sensors. Then, gradients from multiple batches are accumulated before updating the parameters. Combining DAP and DAT, we show how to transform an existing DNN into an adaptive architecture, while keeping the same number of parameters and classification accuracy, and improving the inference time. Besides allowing adaptive sampling rate and sensor selection in a unified solution, DANA also enables power-limited devices to take advantage of convolutional layers' capability in reducing the performed computations according to the dimensions of the sampled data. Experimental results on four datasets show that DANA can maintain classification accuracy in dynamic situations where existing DNNs drop their accuracy, and better generalizes to unseen environments. Most neural network architectures can work with the proposed DANA, without altering other architectural aspects of the original model, while benefiting from the robustness and flexibility DANA provides.

In this chapter, we first discuss the sampling rate and sensor selection problems, and explain how an adaptive neural network can address both problems. Then we elaborate on how a DAP layer works, and how DNNs that use DAP can be efficiently trained using DAT. Finally, we compare DANA with non-adaptive DNNs and other baselines[1].

A.2 Sampling Rate and Sensor Selection

In human activity recognition via mobile and wearable devices, two highly debated characteristics are the appropriate selection of sensors and the sampling rate for the selected sensors. Previous works, which are mostly based on non-deep-learning approaches, have addressed either sensor selection [96, 104, 105, 106, 107, 108] or adaptive sampling rate [109, 110, 111, 112, 113], and usually dedicated a separate classifier for each feasible setting of available sensors [219, 220]. Moreover, the trade-off between classification accuracy and power consumption varies with changes in the sampling rate, the sensor streams used by the classifier, the type of the current activity, and the physical characteristics of the wearer [221]. Power-aware sensor selection may use a meta-classifier [96] or the prediction of future activities from the current one to deselect sensors that are not useful for future activities [104]. Alternatively, a graph model representing the correlation among sensors with a greedy approximation can be used for sensor selection [106]. Finally, a subset of sensors can be dynamically selected by minimizing an objective function that takes

[1]Code and data to reproduce results are publicly available at `https://github.com/mmalekzadeh/dana`

classification accuracy and the number of sensors as inputs [108].

Defining the minimum sampling rate that captures discerning frequency components in different human activities, for example, to minimize power consumption [75], is challenging [110] as the minimum required sampling rate varies across users, activities, and sensor positions. Khan *et al.* [110] show that the minimum sampling rate across activity recognition datasets varies between 22 and 63 Hz, for a 99% Kolmogorov-Smirnov similarity test [126]. Yan *et al.* [221] propose to use a classifier with the highest sampling rate and then, after recognizing the current activity, switch to a lower sampling rate with another classifier and only monitoring whether the current activity changes or not. In order to lower the power consumption, AdaSense [109] uses a lower sampling rate to periodically check for changes in the current activity of the user, and if a change has detected, then it uses a higher sampling rate to classify the new activity. To determine the best trade-off between power consumption and classification accuracy, Cheng *et al.* [111] find an optimal classification model as well as appropriate sampling rates using a continuous state Markov decision process that is only appropriate for training simple classifiers, such as softmax regression, and not applicable in training DNNs.

A.3 Adaptive DNNs

The *architecture* of a DNN is mainly defined by the type and number of the layers, and the way these layers are connected to each other [198]. A commonly used DNN architecture to classify sensor data is composed of a convolutional neural network (CNN) followed by a feedforward neural network (FNN) or a recurrent neural network (RNN) [101, 102, 103, 117, 116, 182, 119, 183, 118]. A CNN is inherently adaptive to input data of variable dimensions and learns features relevant for the classification task. Feeding such extracted features to an FNN or an RNN facilitates generalization, compared to using multi-channel raw sensor data [181]. However, FNNs work on fixed-dimension input data only, whereas RNNs work on a variable number of samples, but on a fixed number of data streams. The main difference between FNNs and RNNs is that there are no feedback connections in an FNN. A feedback connection feeds the output of a neuron in a neural net, to the neuron itself. Every neuron in an RNN includes a feedback connection which helps RNN to capture temporal correlation among consecutive samples.

To address these limitations, a pre-processing stage can be added to the inference pipeline that up/down-samples data to a fixed rate or imputes dummy data (*e.g.* zeros or the average value of the training data) to compensate for missing samples [222], but these solutions reduce classification accuracy and can even raise some security challenges. In [223], authors show that in downsampling data, from higher resolutions to a lower resolution, some discerning patterns in the data are either removed or changed such that the resampled data of a specific class will be categorized into another class. They show how adversaries can take advantage of their knowledge about the used downsampling method to generate adversarial examples for deceiving machine learning models.

Alternatively, one can perform a global maximum or average pooling over all values of each convolutional filter map [224]. A global pooling layer maps each CNN's output of variable dimensions into a one-dimensional vector whose length is equal to the number of the convolutional filters, by taking the maximum or average over each filter map. Global pooling ignores the inherent spatial structure of the data and we show that it can cause accuracy loss.

To mitigate the shortcomings of global pooling in visual object recognition, spatial pyramid pooling (SPP) [225] runs pooling on a pyramid that is created by hierarchically dividing a feature map into equally sized segments. SPP is supported by a weight-sharing mechanism for training CNN-FNN architectures on different image resolutions, and is used for transfer learning: the CNN part trained on the source dataset can be used with other FNNs on the target dataset [226]. SPP has promising results in image processing [227], but is not applicable to CNN-RNN architectures where recurrent layers, such as LSTMs [228], need two-dimensional inputs that preserve the temporal correlation among consecutive samples and across different sensor streams. Therefore, global or pyramid layers are not directly applicable to RNNs, which are often preferred for time-series classification [229]. Moreover, existing pooling layers cannot turn DNNs for multivariate sensor time series adaptive to temporal changes in the input data dimensions, unless the DNN's architecture is considerably changed, which may reduce accuracy.

A DNN that can process variable input dimensions should also be trained to produce accurate outcomes with any feasible input dimensions. *Weight Averaging* [225] was proposed to build a single DNN by averaging the weights across multiple DNNs trained in parallel across different settings, which is also used in federated learning where the goal is to collaboratively train a shared model across different users [144]. Similarly, in meta-learning, the goal is to train a model across a large number of different tasks [230]. *Reptile* [231] is a meta-learning algorithm that uses the average parameters of multiple DNNs, each trained on a different task, as input to the optimizer for updating the DNN's parameters, instead of directly using the gradients of the loss function.

Unlike existing adaptive pooling layers [224, 225], our proposed DAP layer considers the temporal correlations in data as well as the absence of some of the

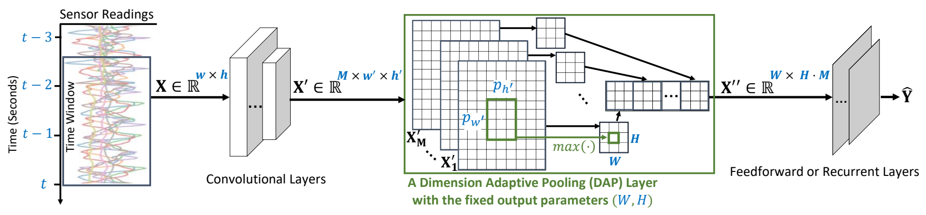

Sensor Readings
Time (Seconds)
Time Window
$t-3$
$t-2$
$t-1$
t
$\mathbf{X} \in \mathbb{R}^{w \times h}$
Convolutional Layers
$\mathbf{X}' \in \mathbb{R}^{M \times w' \times h'}$
$p_{h'}$
$p_{w'}$
$\mathbf{X}'_M$
$\mathbf{X}'_1$
$max(\cdot)$
H
W
A Dimension Adaptive Pooling (DAP) Layer
with the fixed output parameters (W, H)
$\mathbf{X}'' \in \mathbb{R}^{W \times H \cdot M}$
Feedforward or Recurrent Layers
$\hat{\mathbf{Y}}$

Figure A.1: A DAP layer enables a DNN performing classification on input data of variable dimensions. At time t, a window $\mathbf{X}$ of dimensions $w \times h$ is produced. The number of *streams*, h, depends on the number of available sensors, while the number of *samples*, w, per each stream depends on the current sampling rate (in Hz) and the length of time window (in seconds). For example, a 2.5-seconds time window of 50 Hz data generated by accelerometer, gyroscope, and magnetometer of a smartwatch has dimensions of $w = 125$ and $h = 3 \cdot 3 = 9$. Convolutional layers process $\mathbf{X}$, then DAP is applied to the M outputs of the last convolutional layer, $\mathbf{X}'$. DAP uses an adaptive kernel of size $p_{w'} \times p_{h'}$ that is calculated based on DAP's chosen parameters, (W, H), and the dimensions of CNN's outputs, $w' \times h'$. Finally, data of fixed dimensions, $\mathbf{X}''$, is provided for the following feedforward or recurrent layers that are estimating a probability distribution, $\hat{\mathbf{Y}}$, over the possible outcomes. Note that, $\times$ separates the size of each dimension while $\cdot$ denotes product (see the Notations).

sensors. This is particularly important as RNNs are more efficient in processing temporal data than FNNs [102, 118, 117]. Moreover, the proposed DAT resolves the need for weight sharing via training multiple DNNs, providing a faster and more accurate training algorithm, compared to the existing ones [225, 144, 231]. Finally, our solution resolves the need for training and deploying multiple DNNs to perform multimodal fusion [232, 220] which is particularly important for applications running mobile and wearable devices with limited power and processing resources.

A.4 Dimension-Adaptive Pooling

We propose the DAP layer to handle situations where one or more sensors may dynamically be deselected at inference time. The flexibility of DAP aims not only to make DNNs adaptive to changes in the dimension of data, but also to allow efficiently training the DNN such that it provides reliable performance across several combinations of data dimensions.

Considering a multivariate time-series (see Figure A.1), let $\mathbf{X}$ be a time window of dimensions $w \times h$ where w is the number of samples, which depends on the sampling rate and the length of the time window, and h is the number of sensor streams. Particularly in our case, motion sensors have three spatial axes (x, y, z), thus $h = 3 \cdot s$, where s is the number of available sensors. Thus, if a sensor is not available, or deselected, then $\mathbf{X}$ will have 3 fewer streams.

A DNN can cope with inputs of variable dimensions by using a convolutional layer as the input layer. A two-dimensional convolutional layer slides M fixed-sized *filters* across the input data, and computes the dot products between each filter's entries and the data at the current position of the filter, resulting in M two-dimensional filter maps ($\mathbf{X}'$s in Figure A.1). In a stack of convolutional layers, the dimensions of the CNN's output, $w' \times h'$, mainly depend on the dimensions of the input data[2], $w \times h$.

Typically, the CNN's outputs are reshaped into one-dimensional data of size $w' \cdot h' \cdot M$ for FNNs or two-dimensional data $w' \times h' \cdot M$ for RNNs. However, these typical DNNs can cope only with input data of fixed dimensions. One cannot simply use existing layers to make DNNs, which are proposed for processing sensor time-series, adaptive to the sampling rate and sensor selection without imposing any architectural changes. For instance, the single-dimensional data produced by SPP [225] is not appropriate for RNNs where the input must be provided in two dimensions: consecutive samples and parallel streams.

DAP addresses the aforementioned limitations without enforcing assumptions on the DNN architecture to be used. DAP builds upon the global and pyramid pooling ideas and aims to map the outputs of dimensions $w' \times h' \times M$ into data whose dimensions are consistent with the next FNN/RNN layer. The size of the pooling filters in DAP is not limited to be square, hence it generalizes existing adaptive layers [224, 225].

Algorithms A.1 shows the functionality of DAP layer for DNNs processing motion sensor data[3]. Let (W, H) be the pre-specified hyper-parameters for pooling all the M feature maps into an output $\mathbf{X}''$ of single dimension of size $W \cdot H \cdot M$ (if the next layer is FNN) or two dimensions $W \times H \cdot M$ (if the next layer is RNN). DAP first calculates the pooling parameters ($p_{w'} = \lfloor \frac{w'}{W} \rfloor, p_{h'} = \lfloor \frac{h'}{H} \rfloor$) for the received inputs. It then, for every segment of size $(p_{w'}, p_{h'})$ on the received input, chooses the maximum value to create a the output which aims to be of fixed-dimensions (W, H). For larger data dimensions, the pooling parameters adaptively cover a larger segment of the data and for smaller data dimensions the pooling parameters will shrink appropriately. Hence, DAP always produces an output of fixed dimensions.

As an example, consider a CNN-RNN and $s = 3$, $w' = 128$, $h' = 9$, $W = 16$, $H = 3$, $M = 32$. The maximum of every 8 consecutive samples among all 3 axes of each sensor will be chosen for the output, thus producing $\mathbf{X}''$ with dimensions $16 \times 3 \cdot 32$. If we deselect two sensors and choose a sampling rate half of the

[2]Design parameters such as the number of convolutional filters, M, size of the filters, the chosen padding mode, and the stride length of the filters are fixed at inference time.

[3]$\lfloor\rfloor$ denotes floor, $\lceil\rceil$ denotes ceiling, and $\lfloor\rceil$ denotes rounding to the nearest integer.

Algorithm A.1 Dimension Adaptive Pooling.

```
1: Input: $\mathbf{X}'$: filter maps received from a CNN,
   $(W, H)$: the fixed output parameters.
2: Output: $\mathbf{X}''$: input to the next layer.
3: $M, w', h' = dimensions\_of(\mathbf{X}')$
4: $\mathbf{X}'' = \{\}$
5: for $m = 1$ to $M$ do
6:    $\mathbf{V} = \mathbf{X}'[m]$
7:    $\mathbf{Z} = copy\_of(\mathbf{V})$
8:    $a = \mathsf{max}(\lceil (H - h')/3 \rceil, 0)$
9:    for $i = 1$ to $a$ do
10:      $\mathbf{Z} = concatenate\_vertically(\mathbf{Z}, \mathbf{V})$
11:   end for
12:   $\mathbf{Z} = \mathbf{Z}[0 : w', 0 : max(h', H)]$
13:   $p_{w'} = w'/W$
14:   $p_{h'} = h'/H$
15:   $\mathbf{Q} = \{\}$
16:   for $i = 1$ to $W$ do
17:      for $j = 1$ to $H$ do
18:         $r_1 = \lfloor i \cdot p_{w'} \rceil$
19:         $r_2 = \lfloor (i+1) \cdot p_{w'} \rceil$
20:         if a=0 then
21:            $c_1 = \lfloor j \cdot p_{h'} \rceil$
22:            $c_2 = \lfloor (j+1) \cdot p_{h'} \rceil$
23:         else
24:            $c_1 = \lfloor j \cdot \lfloor (a+1) \cdot p_{h'} \rfloor \rceil$
25:            $c_2 = \lfloor (j+1) \cdot \lfloor (a+1) \cdot p_{h'} \rfloor \rceil$
26:         end if
27:         $\mathbf{Q} = append\Big(\mathbf{Q}, max(\mathbf{Z}[r_1 : r_2, c_1 : c_2])\Big)$
28:      end for
29:   end for
30:   $\mathbf{X}'' = append(\mathbf{X}'', \mathbf{Q})$
31: end for
```

original, this means $w' = 64$ and $h' = 3$, then DAP adaptively keeps the output fixed, by choosing the maximum of every $8/2 = 4$ consecutive samples of each axis of the only available sensor.

The input of DAP has dimensions $w' \times h'$, which can be variable at training and inference time; whereas the output of DAP, $\mathbf{X}''$, is fixed and includes $W \cdot H \cdot M$ values. The first inner loop in Algorithm A.1 (lines 9-11) handles situations when one or more sensors are unavailable. Depending on the value of H, we may need to replicate some of the existing streams to satisfy fixed-sized outputs. For a DAP layer with $W = 16$, $H = 9$, and 3 sensors, the possible situations are as follows.

(1) All the sensors are available ($h' = 9$), thus $a = 0$ (in line 8 of Algo-

rithm A.1) and the algorithm skips the loop (Lines 9-11).

(2) One sensor is unavailable ($h' = 6$), thus $a = 1$ and the loop fills the gap by *vertically concatenating* a copy of the data from available sensors to the second dimension of the input data. Therefore $\mathbf{Z}$ will have $h' = 12$ streams and Line 12 truncates $\mathbf{Z}$ on its second dimension to ensure that its second dimension satisfies $H = 9$.

(3) Two sensors are unavailable ($h' = 3$), thus $a = 2$ and the algorithm fills the gap by *vertically concatenating* a copy of the data from the only available sensor two times. So, we will again satisfy $H = 9$.

Note that Line 12 guarantees that, in case of concatenating data to $\mathbf{Z}$, the number of streams in data never exceeds H, and it has a neutral effect when $h' = H$. The next two loops, in Lines 16-29, perform the pooling operation over the consecutive segments of $\mathbf{Z}$.

Following our example, in each iteration we consider a segment of data including w'/W samples and h'/H streams. For example, if $w' = 96$ then the maximum of every 6 consecutive samples of each sensor stream (*e.g.* the accelerometer's x-axis) is calculated and appended to $\mathbf{Q}$ (Line 27). If we only change H to 3, instead of 9, a will be zero, thus no concatenation happens. On the other hand, every segment of data includes $6 \cdot 3 = 18$, $6 \cdot 2 = 12$, or $6 \cdot 1 = 6$ samples if all three, two, or only one sensor(s) are (is) available, respectively. Therefore, DAP adaptively changes the size of pooling segments to ensure that the output dimensions are fixed. Also note that, in Lines 16-29, the first loop runs on the first dimension W and the second loop on the second dimension H: this order preserves the temporal correlation between consecutive samples, which is necessary for DNNs that use recurrent layers.

A.5 Dimension Adaptive Training

Although a DNN using DAP can accept input of any dimensions, it should be properly trained for being adaptive to changes, otherwise, its classification accuracy will be low when the input's dimensions change. The *multi-size training* procedure [225] for images trains in parallel two DNNs with different dimensions but with shared weights. This multi-size training is problematic with sensor data because of the large variety of possible training situations. For example, 10 sampling rates (e.g. 5, 10, 15, ..., 50 in Hz) and the 7 possible combinations of 3 sensors lead to 70 DNNs to train.

To overcome this challenge and provide a reasonable, converging training strategy, we propose dimension adaptive training (DAT). In DAT, we train a single DNN, and therefore there is neither a need for weight sharing nor for choosing a specific set of data dimensions. DAT comprises of two main

Algorithm A.2 Dimension Adaptive Training.

```
1: Input: $\mathbb{D}$: training datasets, $\Theta$: trainable parameters, $E$: number of epochs,
   $\mathbb{U}$: a subset of all feasible dimensions, $B$: number of batches used in each
   optimization round, $K$: the size of each batch.
2: Output: $\Theta$: optimized parameters.
3: for $e = 1$ to $E$ do
4:   while not feeding the DNN all data in $\mathbb{D}$ do
5:     $\mathcal{G} = 0$
6:     for $b = 1$ to $B$ do
7:       $\mathbb{X} = generate\_random\_batch(\mathbb{D}, K)$
8:       $\mathbb{X} = dimension\_randomization(\mathbb{X}, R)$
9:       $\hat{\mathbb{Y}} = forward\_pass(\Theta, \mathbb{X})$
10:      $\mathcal{G} = \mathcal{G} + Gradients(Loss(\mathbb{Y}, \hat{\mathbb{Y}}))$
11:    end for
12:    $\Theta = optimize(\Theta, \mathcal{G})$
13:  end while
14: end for
```

ideas: *dimension randomization* and *optimization with accumulated gradients* (Figure A.2). The DAT process works by training the DNN on input data of several randomly selected dimensions. For efficiency, the data is processed in batches of input data that has the same dimension. The details of DAT are shown in Algorithm A.2.

Each round of optimization includes two steps. First, a random batch, $\mathbb{X}$, of K time windows of the highest available sampling rate is generated from the dataset. For batch number, b, the available sensors are chosen randomly and a random sampling rate is chosen with data downsampled using bilinear interpolation. All the time windows in batch b have the same dimensions[4] $w_b \times h_b \in \mathbb{U}$. Considering an example where the possible sampling rates range from 6 Hz to 50 Hz, instead of considering all the 45 possible cases, we only choose a subset $\mathbb{U}$, for instance including 8 sampling rate {6, 12, 18, 25, 31, 37, 43, 50} (in Hz). In the epoch e, *dimension_randomization*() in Line 8 randomly and uniformly chooses, for instance $B = 4$ of this 8 sampling rates without replacement. Second, a forward pass is performed on batch b giving a vector of predictions, $\hat{\mathbb{Y}}$. we calculate the average loss value (*e.g.* categorical cross-entropy) of the predictions compared with the true labels, $\mathbb{Y}$, and its gradients, corresponding to the Θ, is accumulated into $\mathcal{G}$. These two stages are repeated B times, then using the accumulated gradients, $\mathcal{G}$, the parameters the DNN, Θ, are updated at once.

As DNNs tend to forget previously learned information upon learning from

[4]Note that, the key point of efficiently training DNNs on GPUs is in eliminating loops by matrix multiplications that forces all samples in each batch, b, to have the same dimensions in a forward pass.

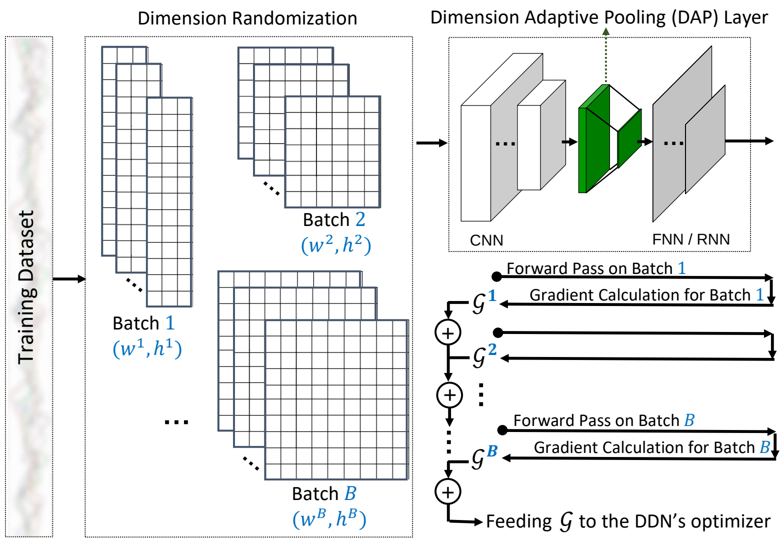


Figure A.2: Overview of each iteration in DAT. B batches of time windows are randomly generated from a training dataset. Samples within each batch have the same dimension while samples among batches have different dimensions. All time windows in each batch are transformed into the same randomly chosen dimensions $w^b \times h^b$. Every batch is iteratively fed into DNN and the corresponding gradients with respect to the current loss value are accumulated into $\mathcal{G}$. Finally, the parameters of the DNN are updated based on $\mathcal{G}$.

new data, updating Θ immediately after computing losses for each batch of data causes catastrophic forgetting [233], in which the DNN may repeatedly forget how to perform classification on the previously trained dimensions. We show that the combination of dimension randomization and gradient accumulation not only helps DAT to prevent catastrophic forgetting but also to converge better.

It is worth noting that the gradient accumulation in DAT is similar but a different concept than the typical method of keeping track of gradient *momentum* in stochastic gradient descent [234]. In a momentum-based optimization, a scaled version of the gradients used in the previous round is added to the gradients of the current round. Thus, at each round, the DNN's parameters are updated based on the current gradients and the history of the past gradients. On the other hand, DAT is a procedure for computing the required gradients in a single round, hence we can use DAT and a momentum-based optimization, such as Adam [235], together.

A.6 Evaluation

We evaluate DANA on four public datasets of human activity recognition: *UCI-HAR*, *UTwente*, *MobiAct*, and *MotionSense* (see Table 2.2). *UCI-HAR* is a widely used dataset [101, 102, 103, 117], but only including two sensors and 6 activities. Thus, we also use *UTwente* which includes three sensors and 13 activities. *MobiAct* [175] and our *MotionSense*, both include accelerometer and gyroscope collected by a smartphone in the pocket of the users' trousers. We use the data from the 6 activities in common among MobiAct and MotionSense, thus we can use the entire MotionSense dataset for test purposes and do not use it in the training. For all experiments, we use a time window $T = 2.56$ seconds (i.e. a maximum of 128 samples per window) [101, 103, 102].

In the following, we show how to transform three state-of-the-art CNN-FNN/RNN architectures for sensor-based human activity recognition [101, 103, 102] into a DANA, and discuss how DAP works with and without using DAT. We also evaluate the advantages of DANA at training time, compared with three other training procedures: *standard*, *weight averaging* [225], and *Reptile* [231], and at the inference time compared with two alternatives baselines: *imputation* and *resampling* with and without *data augmentation*. Finally, we perform a cross-datasets experiment to show the generalization of DANA.

A.6.1 Transforming a DNN into DANA

Figure A.3 compares the original architectures proposed in the related work [101, 103, 102] and our modified version using DAP. A 2D convolutional layer is shown by Conv2D$(n, (k_1, k_2), padding, activation)$ where n is the number of neurons, (k_1, k_2) is the size of the 2D kernel used by each neuron; if *padding* is *same*, the output and the input of the layer have the same dimensions, otherwise if it is *valid* then the output has k_1-1 and k_2-1 values fewer than the number of inputs in the first and second dimensions, respectively; *activation* shows the nonlinear activation function applied to the output of the layer; Dense$(n, activation)$ shows a fully-connected feedforward layer; LSTM$(n, activation)$ shows a recurrent LSTM [228] layer; MaxPool2D$((p_1, p_2))$ shows a 2D maximum pooling layer that outputs the maximum value of each window including $p_1 \cdot p_2$ data points; Dropout(q) applies Dropout [203] with probability q that randomly sets the output of a neuron to 0 during each forward pass in training time; FlattenXD() reshapes its inputs into 1D or 2D outputs.

To transform a non-adaptive DNN into a DANA, we only use a single DAP layer, instead of "maximum pooling" layers, with an appropriate pooling parameter (W, H), and instead of "valid" padding, we use "same" padding. Thus, we can keep the total number of trainable layers and parameters exactly the

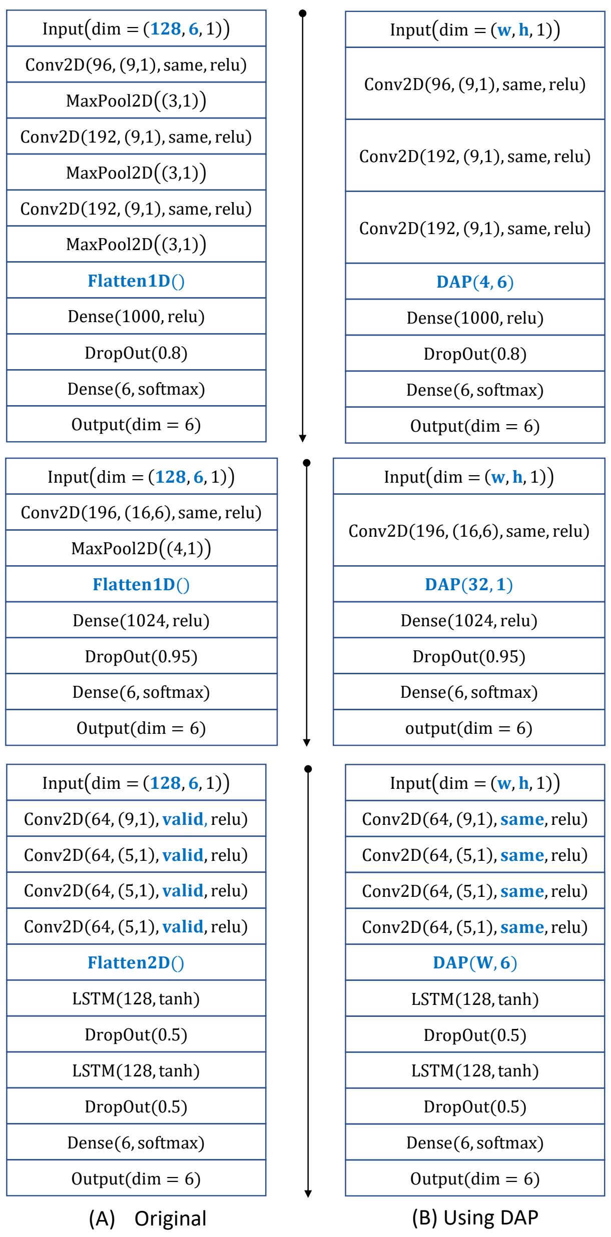


Figure A.3: The original (left) DNNs proposed by [101] (top), [103] (middle), and [102] (bottom) versus their DANA version (right). The differences between each pair of DNNs are shown in **blue bold** font. We use the TensorFlow [202] naming conventions.

same as the original DNN.

Table A.1: Classification accuracy of three benchmark DNNs on UCI-HAR dataset, the original non-adaptive model, versus the corresponding adaptive model with a DAP layer.

				Accuracy (%)	
Architecture	Size	DNN $[(W, H)]$	Setting	Mean±STD	Maximum
CNN-FNN	5,114,014	[101]	*validate*	93.85±.49	94.51
			test	91.92±.97	93.14
		with DAP(4,6)	*validate*	93.73±.84	95.39
			test	91.88±1.1	93.04
CNN-FNN	6,448,714	[103]	*validate*	93.80±.59	94.78
			test	92.75±.73	94.02
		with DAP(32,1)	*validate*	93.89±.41	94.57
			test	92.57±.60	93.65
CNN-RNN	457,030	[102]	*validate*	94.13±.53	95.25
			test	91.66±1.3	93.45
		with DAP(4,6)	*validate*	94.69±.31	95.32
			test	94.07±.39	94.77
		with DAP(8,6)	*validate*	94.82±.34	95.32
			test	93.47±.81	94.33
		with DAP(16,6)	*validate*	94.69±.48	95.18
			test	93.19±.59	94.02
		with DAP(32,6)	*validate*	94.74±.33	95.32
			test	93.23±.92	94.70

A.6.2 Original DNNs versus Their DANA Version

To ensure the comparison is fair among all DNNs, we use the same number of epochs (1,000), early stopping patience (100 epochs), and batch size (128). We also use the same optimizer as reported in the corresponding works: for the models with FNN, Adam [235], and for the models with RNN, RMSProp [236]. We run each model 10 times and report the mean and standard deviation for classification accuracy.

Table A.1 compares the classification accuracy for each DNN architecture with that architecture using DAP instead of their standard maximum pooling. Although we set (W, H) such that the transformed architecture has the same model size as the original DNN; it is possible to choose any other values getting a larger or smaller size DNN. Here, two settings are considered for evaluations. (i) *Validate*: where the whole training dataset is used for training, and the test dataset is used for validation. Thus, each DNN is trained on the training dataset and the validation dataset is used to check the accuracy of the model after each epoch. This is the setting that is used by other works re-implemented [101, 103]. (ii) *Test*: where 10% of the training dataset is randomly chosen as the validation set and the rest (90%) is used as the training set. Here, the test dataset is used to evaluate the best trained DNN on the validation set.

Table A.1 shows that for all three architectures the classification accuracy

Table A.2: Classification accuracy (%) of three benchmark DNNs on UTwente, the original non-adaptive model versus the corresponding adaptive model using a DAP layer.

			Accuracy (%)	
DNN	Size	Ref.	Mean±STD	Maximum
CNN-FNN	7,425,021	[101]	59.20±9.6	76.99
	12,878,417	[103]	78.85±6.2	88.01
CNN-RNN	556,237	[102]	93.68±.36	94.55
		with DAP(8,9)	94.35±.37	95.16
		with DAP(16,9)	94.64±.40	95.40

of the adaptive version is either slightly improved or almost the same as the original DNN, in both *validate* and *test* setting. This suggests that the DAP layer does not lead to a loss in accuracy while providing the desirable adaptivity. The CNN-RNN model has 10 times fewer parameters than the FNN but has a better classification accuracy. The CNN-RNN with DAP significantly improves accuracy in the *test* setting, thus suggesting that DAP helps the model to better generalize to the test data. RNNs are also flexible in their first dimension, which is the number of samples per stream. Thus, unlike FNNs, we can use DAP layers having different pooling parameters for the first dimension while keeping the rest of the architecture the same. Changing the pooling parameters leads to slightly different accuracies, which can be used as part of a fine-tuning pipeline for DNN models but would not be possible with FNNs.

To see the performance of the DNNs and DANA on another dataset, we evaluate existing DNNs (without fine-tuning) on UTwente, which is a different dataset from those datasets used in the original papers of the implemented DNNs. Table A.2 shows that the CNN-RNN architecture has better generalization than the other two CNN-FNN architectures. The model using DAP maintains a comparable accuracy.

We see that transforming a DNN into DANA does not change the trainable parameters of the DNN, whereas using other global pooling or SPP layers would require a change in the size of the FNN/RNN layers and consequently the number of trainable parameters of the original DNN. As an experiment, to measure the effect of using the global average pooling layer [224], we run a similar experiment on CNN-RNN [102] with UCI-HAR. The model size is reduced from 457K to 293K, but also the classification accuracy (%) reduced from $94.13 \pm .53$ to $78.6 \pm .81$.

It should be noted that this experiment only aims to show that using a DAT layer does not degrade the accuracy, and not to show that by using a DAT layer we can achieve better accuracy in a fixed-dimensions scenario. As

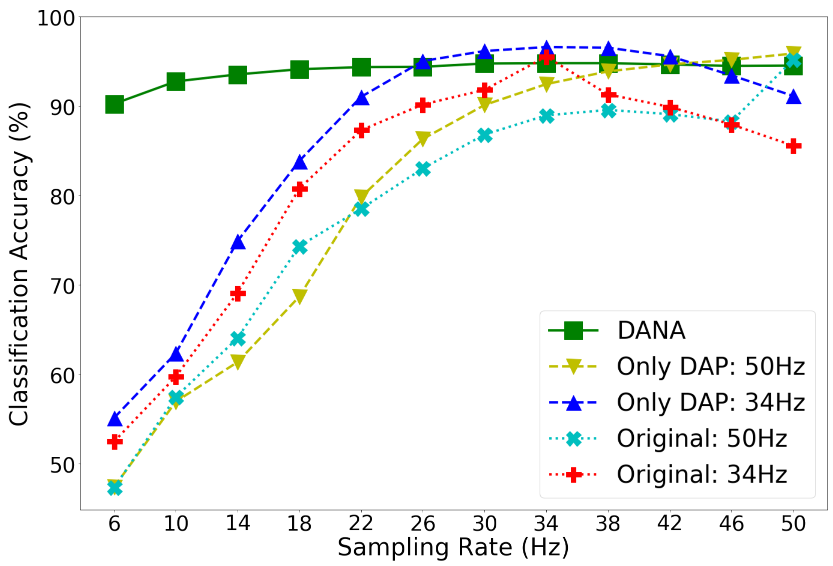


Figure A.4: Classification accuracy of CNN-RNN [102] on UCI-HAR for different sampling rates.

we show in other experiments, the main advantage of DANA is keeping high accuracy in variable-dimensions scenarios. As the *validate* setting is what is considered in other works [101, 103], where they report the *best accuracy* that corresponding DNN can achieve on the dataset, for the rest of the experiment we use the *validate* setting for fair comparisons. Note that as we do not tune any hyper-parameters and do not change the size of trainable parameters in all DNNs, similar relative results are achieved in the *test* setting. But, unlike the *validate* setting which has only one instance (*i.e.* a training set and a validation set), the *test* setting could be biased because there are many possible instances depending on the randomly chosen 10% validation set.

Figure A.4 compares the original and DANA version of the DNN proposed by [102], as CNN-RNN performs better. First, using the original DNN, the model can only be trained and validated on data of fixed dimensions. To see how the original model performs if we change the sampling rate, we keep the weights and parameters of the original model and use them on a version using DAP (lines with a cyan cross and red plus). Second, we have a DNN version that only uses DAP during the training but only trained on a fixed sampling rate (lines with yellow and blue triangles). Finally, we have the DANA version (the line with green square) which not only uses DAP, but it also uses DAT with $B = 4$ and $\mathbb{U}$={6, 12, 18, 25, 31, 37, 43, 50 } (in Hz).

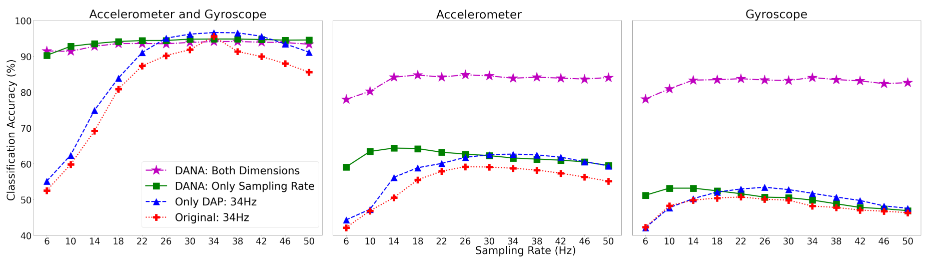


Figure A.5: Classification accuracy of CNN-RNN [102] on UCI-HAR with a variable number of sensors.

A.6.3 Changes in Sampling Rate and Sensor Availability

Although DANA version slightly reduces the accuracy on the specific sampling rates the other DNN versions were trained on (34Hz and 50Hz), it considerably outperforms the other DNN versions across the whole range of sampling rates, where the accuracies of other DNNs drops quickly when moving away from their pre-defined settings. Because the best accuracy of DANA occurs at 34 Hz, for this reason, we trained other DNNs on this specific sampling rate, thus suggesting that DAT is consistent in its performance while being adaptive.

To see the effect of sensor selection, Figure A.5 shows the replication of previous experiments (in Figure A.4) in different sensor-selection scenarios. Here, DANA is trained to account for the possibility of deselecting (or missing) sensors, and the other lines are the same as in Figure A.4 where the training only considers sampling rate selection and does not account for sensor selection. During training, 50% of the time, DANA is trained on both sensors, and 50% of the time with one of accelerometer or gyroscope (randomly chosen with equal probability). The values of B and $\mathbb{U}$ are the same as Figure A.4.

Figure A.5 (left) shows that the adaptivity to the sensor selection is not associated with a large penalty when all sensors are present. Figure A.5 (middle) shows that when the gyroscope is deselected, DANA maintains its accuracy around 85% while the accuracy of other DNNs falls rapidly to around 60%. Similarly, Figure A.5 (right) shows that when the accelerometer is deselected the accuracy for DANA remains around 85% while other DNNs fall to 50% or less. It is interesting that while for other DNNs deselecting accelerometer data causes more accuracy loss than deselecting gyroscope, the type of the deselected sensor has a reduced effect on DANA.

In Figure A.6 we make the CNN-RNN model adaptive to both sampling rate and sensor availability using DAP and DAT on UTwente, and we achieve similar accuracy to the original model. Note that, while original DNNs are not reliable when the data dimensions change, DANA provides at least 55% accuracy across

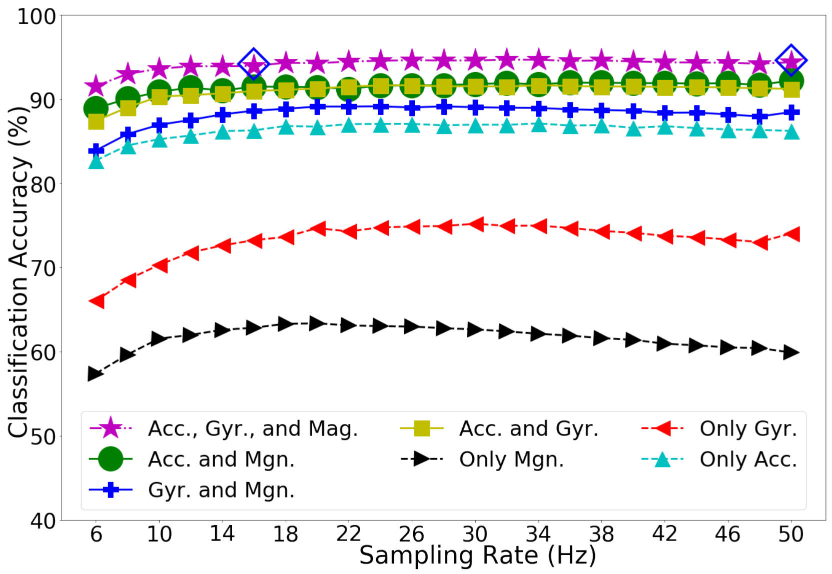


Figure A.6: Classification accuracy on UTwente with the DANA version of the CNN+RNN proposed by [102]. The two points shown by ◇ at 16 Hz and 50 Hz shows the accuracy of the original DNN when trained on these fixed sampling rates.

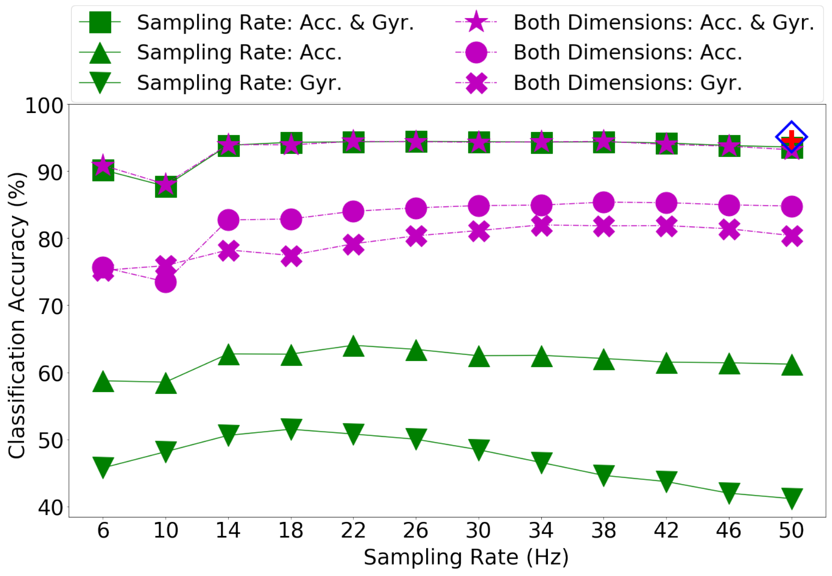


Figure A.7: Classification accuracy on UCI-HAR using a CNN-FNN [101]. The data points + and ◇ at 50 Hz refer to Table A.1 accuracies.

7·45=315 feasible data dimensions.

Figure A.7 compares the accuracy for different versions of a CNN-FNN proposed by [101]. The accuracy of DANA, trained on both variable sampling rates

Table A.3: Accuracy (%) of CNN-RNN architecture proposed in [102] trained on the MobiAct training dataset and validated on the MobiAct and MotionSense test datasets.

DNN	MobiAct	MotionSense	
		Normalized	Pseudo-Normalized
Original	98.64	69.87	44.16
Only DAP	98.91	74.65	49.65
DANA	98.18	74.60	47.97

and variable sensors, remains high for a variety of sampling rates. When one of two sensors is deselected, the accuracy loss is much smaller than the one obtained from training only on variable sampling rates, with fixed sensors. The accuracy loss for this generalization is small when compared with the single points that show the situations where the DNN is only trained on a fixed sampling rate with all sensors present without and with DAP, respectively.

A.6.4 DANA Across Datasets

We train three versions of the CNN-RNN proposed in [102] on MobiAct and test them on MotionSense. We follow two settings: *normalized*, where we normalize the MotionSense data to zero mean and unit standard deviation, and *pseudo-normalized*, where we use the MobiAct statistics to normalize the MotionSense data to mean zero and unit standard deviation. The latter is standard practice as, when streaming data, the mean and standard deviation are not known in advance. All three models have the same number of parameters and were trained under the same training setting.

Table A.3 shows that DANA generalizes better on the test dataset, in terms of accuracy, by about 5 percentage points. This result confirms that for the corresponding datasets, using DAT to make a DNN adaptive helps to achieve a more accurate model in an unseen environment[5]. Although the DANA version has a slightly lower accuracy than the one using only DAP, DANA is the only reliable model when the dimensions of the input data change (see Figure A.8). Thus, while DANA shows a bit smaller accuracy than the original DNN on MobiAct, this will considerably pay off when DANA is used in dynamic settings which we have shown in the results of the previous experiments.

[5]MobiAct was collected with a Samsung Galaxy S3 in the right or left pocket of the trousers. MotionSense was collected with an iPhone 6s in the front, right pocket of their trousers.

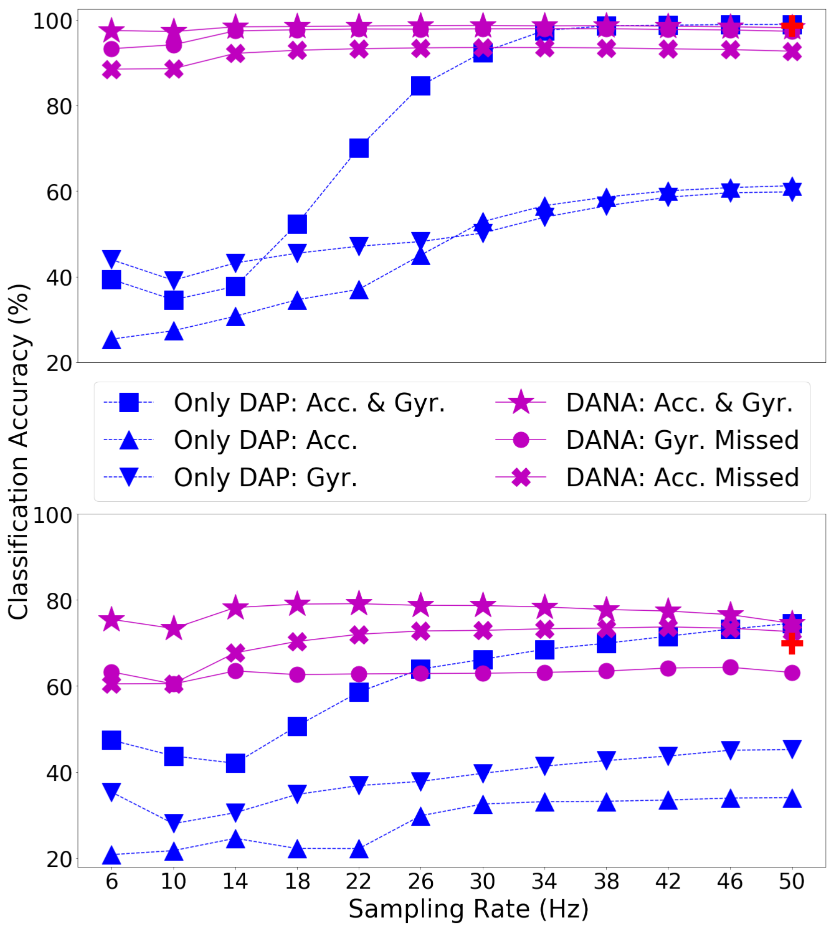


Figure A.8: Classification accuracy of a CNN+RNN [102] trained on MobiAct [175] and tested on (top) MobiAct test set and (Bottom) our MotionSense dataset. The single data point shown by + at 50 Hz refers to the accuracy achieved with the original DNN in Table A.3.

A.6.5 Comparison with Baselines

An alternative baseline to DANA is to take the original model and every time the sampling rate is changed, we fix the problem by re-sampling data to the original sampling rate, and if some sensors are missing or deselected, we impute dummy data, *e.g.* zeros. Another baseline is to use an *augmented* training dataset by making a copy of the data for every possible combination of sensors availability. For example, with 3 sensors, we build 6 other samples for each sample time-window in the training dataset, one for each possible setting. Hence, the training time for the *augmented* case is 7 times the training time for the original and DANA.

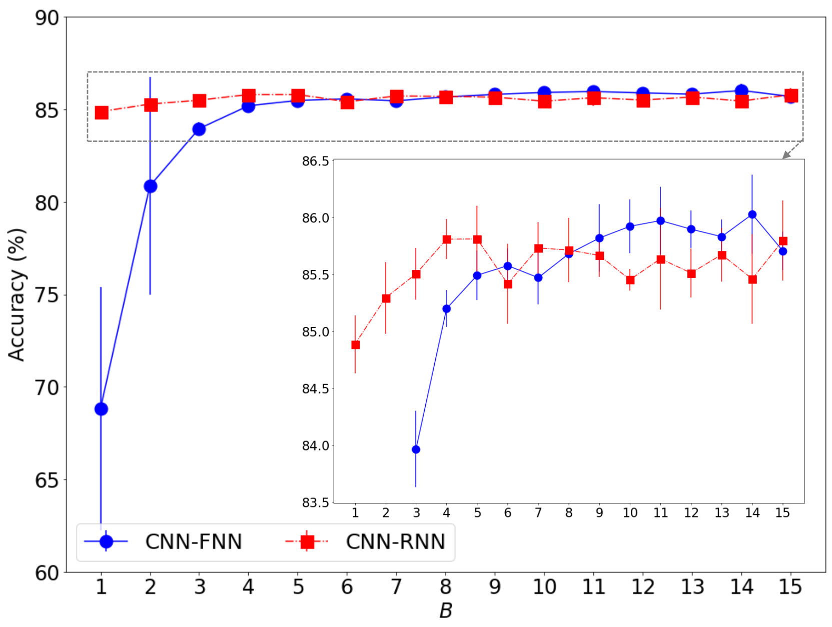


Figure A.9: Effect of the size of resampling batches B on UCI-HAR with CNN-RNN [102]. Each point is the average and each segment shows the standard deviation for 5 runs.

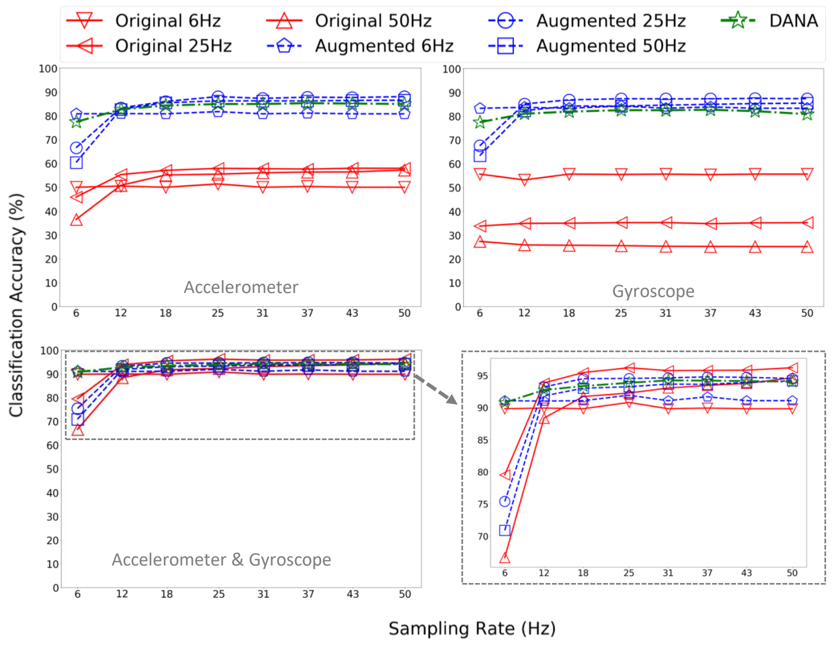


Figure A.10: Comparing baselines with DANA on UCI-HAR with CNN-RNN [102].

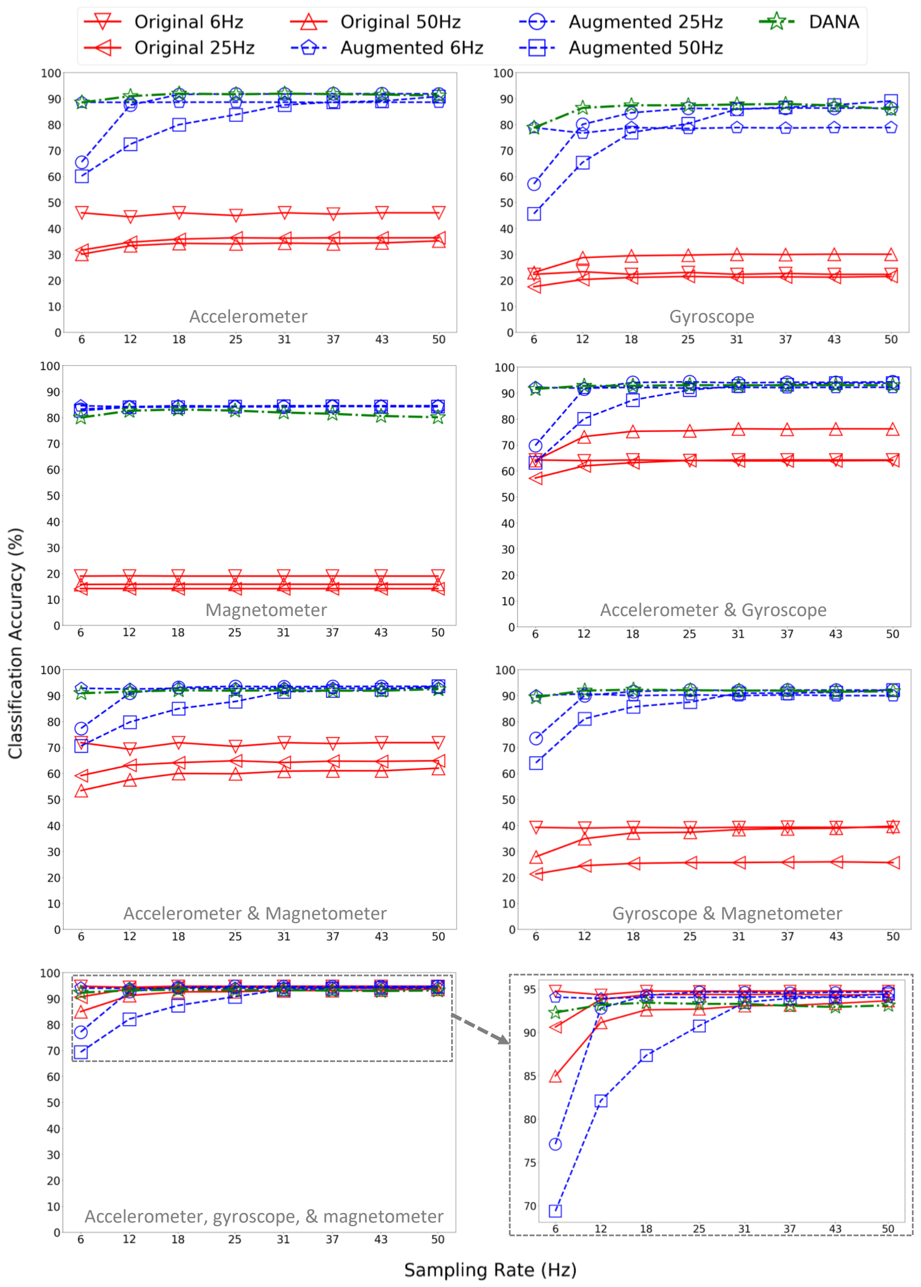


Figure A.11: Comparison of baselines and DANA with CNN-RNN on UTwente [102].

Table A.4: Comparison of different training methods for turning a DNN adaptive to changes in the input data dimensions. Acc denotes the classification accuracy and times (average time per each training epoch) are in the unit of seconds.

Dataset	UCI-HAR [100]						UTwente [99]	
DNN Model	CNN+FNN [101]		CNN+RNN [102]				CNN+RNN [102]	
Optimizer	Adam		RMSProp		Adam		Adam	
Method	**Acc**	**Time**	**Acc**	**Time**	**Acc**	**Time**	**Acc**	**Time**
Standard	69.94±6.08	4.24±.29	86.21±.11	6.03±.17	86.98±.21	5.55±.15	87.02±.51	14.24±.44
WeightAvg	85.90±.25	5.63±.13	86.61±.19	6.54±.10	87.02±.21	6.07±.14	88.14±.99	15.03±.45
Reptile	85.73±.25	5.50±.11	15.91±2.25	6.23±.10	86.87±.26	5.96±.19	87.87±.74	15.01±.43
DAT (Ours)	85.74±.20	3.68±.08	87.12±.13	5.58±.15	87.50±.11	5.43±.15	88.91±.59	13.30±.57

Figure A.10 and A.11 compares DANA with these baselines. We see that DANA can train a single model that outperforms the original model across all 56 settings, and also outperforms the augmented case for lower sampling rates (while remaining competitive for larger sampling rates). Note that the augmented baseline needs longer training time and also does not reduce the computations when sampling rate changes or some sensors are unavailable or deselected. These results show that DANA can capture the correlation, or information redundancy, between different sensor streams. For example, when the original models miss a sensor or two, their accuracies considerably fall, because they are not able to substitute the missed information using the available ones. But DANA keeps the accuracy at the same level as the augmented one, while it does not need to endure the difficulties of the augmented one at the training and inference time.

A.6.6 Comparison with other Training Approaches

We explore the impact of the value of the training hyper-parameter B on the accuracy. Figure A.9 shows, for both CNN-FNN and CNN-RNN, the classification accuracy is comparable for RNN for $B \geq 4$ (whereas FNN cannot be trained with these values), and in general, it suggests setting a $B \geq 5$.

Table A.4 shows that DAT either outperforms or achieves comparable accuracy to other methods in several settings, and is always faster in training. The training approaches we compare with are *Standard*, the typical training procedure that includes a forward pass on a batch of data to calculate loss value, following with a backward pass to update model parameters based on the gradients; *WeightAvg* [225, 144], when each of the B copies of the model is trained on a dedicated batch of data using the *Standard* approach, and then the average parameter values of updated B copies will be used to update the central model; and *Reptile* [231], which performs the same procedure as *WeightAvg*, except the last step when the average parameters of B copies, each trained on a different batch of data, is fed to the chosen DNN optimizer, instead of the typical gradients of the loss function, to update the central model. The value of B is fixed to 5 for all experiments in Table A.4. The *time* shows the average training time per each iteration over the whole training dataset (*i.e.* one epoch). Note that, DAT is not sensitive to the chosen optimizer, or dataset, or DNN architecture. Also, Reptile works with Adam (that is mentioned in the original paper [231]), but it cannot be useful with the RMSProp optimizer.

A.7 Summary

We presented DANA, a solution to make deep neural networks adaptive to changes in the dimensions of the input data to cope with adaptive sampling and sensor selection. DANA provides a single trained model that retains high classification accuracy across a range of settings, thus avoiding the need for a separate classifier for each setting at inference time. DANA imposes no limitations on the type of DNN and is flexible in shaping the DNNs without adding or removing trainable parameters. We showed that our proposed approach outperforms the state-of-the-art over a range of sampling rates and retains accuracy when some sensors are unavailable at inference time. For instance, on a dataset of 3 sensors and 13 activities, DANA keeps classification accuracy similar to the original DNN in a range of 6Hz to 50Hz and its accuracy only falls from 95% to around 90% and 85% in case of missing one or two of the three sensors, respectively, while the original DNN cannot handle these changes, or achieve at most 75% and 55% accuracy with resampling and imputation prepossessing.

To cover the whole range of possible data dimensions at inference time, we randomly cover a subset of the possible situations at each round of training. As future work, one can study the behavior of this randomized selection alongside the gradient accumulation procedure to see if it is possible to provide a more theoretical analysis of the convergence and learning. Moreover, the focus and evaluation of DANA was on motion sensor data. It might be helpful to apply DANA to other types of temporal data, such as audio streams or other types of sensors that have different data characteristics, and analyze the effect of becoming adaptive to temporal changes in other data types.

Bibliography

[1] Janice Y Tsai, Serge Egelman, Lorrie Cranor, and Alessandro Acquisti. The effect of online privacy information on purchasing behavior: An experimental study. *Information systems research*, 22(2):254–268, 2011.

[2] Hamed Haddadi, Pan Hui, Tristan Henderson, and Ian Brown. Targeted advertising on the handset: Privacy and security challenges. In *Pervasive Advertising*, pages 119–137. Springer, 2011.

[3] Gary Bente, Odile Baptist, and Haug Leuschner. To buy or not to buy: Influence of seller photos and reputation on buyer trust and purchase behavior. *International Journal of Human-Computer Studies*, 70(1):1–13, 2012.

[4] Andrew Raij, Animikh Ghosh, Santosh Kumar, and Mani Srivastava. Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 11–20, 2011.

[5] Michael Kearns and Aaron Roth. *The Ethical Algorithm: The Science of Socially Aware Algorithm Design*. Oxford University Press, 2019.

[6] Claus-Peter H Ernst and Alexander W Ernst. The influence of privacy risk on smartwatch usage. In *Proceedings of the 22nd Americas Conference on Information Systems*, pages 1–10, 2016.

[7] Emmanuel Sebastian Udoh and Abdulwahab Alkharashi. Privacy risk awareness and the behavior of smartwatch users: A case study of indiana university students. In *Future Technologies Conference*, pages 926–931. IEEE, 2016.

[8] Andrew Hilts, Christopher Parsons, and Jeffrey Knockel. Every step you fake: A comparative analysis of fitness tracker privacy and security. *Open Effect Report*, 76(24):31–33, 2016.

[9] Vivian Genaro Motti and Kelly Caine. Users' privacy concerns about wearables. In *International Conference on Financial Cryptography and Data Security*, pages 231–244. Springer, 2015.

[10] Jan Henrik Ziegeldorf, Oscar García Morchon, and Klaus Wehrle. Privacy in the internet of things: threats and challenges. *Security and Communication Networks*, 7(12):2728–2742, 2014.

[11] GDPR. Data protection and online privacy. `https://europa.eu/youreurope/citizens/consumers/internet-telecoms/data-protection-online-privacy/`, 2018. Accessed: 01-08-2020.

[12] Alan F Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

[13] Tiancheng Li and Ninghui Li. On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–526, 2009.

[14] Benjamin CM Fung, Ke Wang, Rui Chen, and Philip S Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):1–53, 2010.

[15] Yang Xu, Tinghuai Ma, Meili Tang, and Wei Tian. A survey of privacy preserving data publishing using generalization and suppression. *Applied Mathematics & Information Sciences*, 8(3):1103, 2014.

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

[17] Erik Reinertsen and Gari D Clifford. A review of physiological and behavioral monitoring with digital sensors for neuropsychiatric illnesses. *Physiological measurement*, 39(5):05TR01, 2018.

[18] Matthew Fredrikson and Benjamin Livshits. Repriv: Re-imagining content personalization and in-browser privacy. In *2011 IEEE Symposium on Security and Privacy*, pages 131–146. IEEE, 2011.

[19] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[20] Sun-Kyong Hong, Kuldeep Gurjar, Hea-Suk Kim, and Yang-Sae Moon. A survey on privacy preserving time series data mining. In *3rd International Conference on Intelligent Computational Systems*, pages 44–48, 2013.

[21] Giuseppe D'Acquisto, Josep Domingo-Ferrer, Panayiotis Kikiras, Vicenç Torra, Yves-Alexandre de Montjoye, and Athena Bourka. Privacy by design in big data: an overview of privacy enhancing technologies in the era of big data analytics. *arXiv preprint arXiv:1512.06000*, 2015.

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

[23] Sarat Kumar Chettri and Bhogeswar Borah. On analysis of time-series data with preserved privacy. *Innovations in Systems and Software Engineering*, 11(3):155–165, 2015.

[24] Tristan Allard, Georges Hébrail, Florent Masseglia, and Esther Pacitti. Chiaroscuro: Transparency and privacy for massive personal time-series clustering. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 779–794, 2015.

[25] Fabian Laforet, Erik Buchmann, and Klemens Böhm. Individual privacy constraints on time-series data. *Information Systems*, 54:74–91, 2015.

[26] Hao Wang and Zhengquan Xu. CTS-DP: Publishing correlated time-series data via differential privacy. *Knowledge-Based Systems*, 122:167–179, 2017.

[27] Shuang Song and Kamalika Chaudhuri. Composition properties of inferential privacy for time-series data. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 814–821. IEEE, 2017.

[28] Aiden Doherty, Dan Jackson, Nils Hammerla, Thomas Plötz, Patrick Olivier, Malcolm H Granat, Tom White, Vincent T Van Hees, Michael I Trenell, Christoper G Owen, et al. Large scale population assessment of physical activity using wrist worn accelerometers: The UK biobank study. *PloS one*, 12(2):e0169649, 2017.

[29] Rui Wang, Min S. H. Aung, Saeed Abdullah, Rachel Brian, Andrew T. Campbell, Tanzeem Choudhury, Marta Hauser, John Kane, Michael Merrill, Emily A. Scherer, Vincent W. S. Tseng, and Dror Ben-Zeev. Crosscheck: Toward passive sensing and detection of mental health changes in people with schizophrenia. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, page 886–897. ACM, 2016.

[30] Kleomenis Katevas, Hamed Haddadi, Laurissa Tokarchuk, and Richard G. Clegg. Walking in sync: Two is company, three's a crowd. In *Proceedings of the 2nd Workshop on Workshop on Physical Analytics*, WPA '15, Florence, Italy, pages 25–29. ACM, 2015.

[31] Katrin Hänsel, Kleomenis Katevas, Guido Orgs, Daniel C Richardson, Akram Alomainy, and Hamed Haddadi. The potential of wearable technology for monitoring social interactions based on interpersonal synchrony. In *Proceedings of the 4th ACM Workshop on Wearable Systems and Applications*, pages 45–47. ACM, 2018.

[32] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):1–33, 2014.

[33] Muhammad Irfan, Laurissa Tokarchuk, Lucio Marcenaro, and Carlo Regazzoni. Anomaly detection in crowds using multi sensory information. In *15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[34] David C Mohr, Mi Zhang, and Stephen M Schueller. Personal sensing: understanding mental health using ubiquitous sensors and machine learning. *Annual review of clinical psychology*, 13:23–47, 2017.

[35] Katrin Hänsel, Romina Poguntke, Hamed Haddadi, Akram Alomainy, and Albrecht Schmidt. What to put on the user: Sensing technologies for studies and physiology aware systems. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2018.

[36] Yunyoung Nam, Yeesock Kim, and Jinseok Lee. Sleep monitoring based on a tri-axial accelerometer and a pressure sensor. *Sensors*, 16(5):750, 2016.

[37] Zhenyu Chen, Mu Lin, Fanglin Chen, Nicholas D Lane, Giuseppe Cardone, Rui Wang, Tianxing Li, Yiqiang Chen, Tanzeem Choudhury, and Andrew T

Campbell. Unobtrusive sleep monitoring using smartphones. In *7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, pages 145–152. IEEE, 2013.

[38] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization. *ACM Transactions on Internet Technology*, 3(1):1–27, 2003.

[39] Saikat Guha, Bin Cheng, Alexy Reznichenko, Hamed Haddadi, and Paul Francis. Privad: Rearchitecting online advertising for privacy. *Proceedings of Hot Topics in Networking (HotNets)*, 2009.

[40] Peter Brusilovski, Alfred Kobsa, and Wolfgang Nejdl. *The adaptive web: methods and strategies of web personalization*, volume 4321. Springer Science & Business Media, 2007.

[41] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[42] Michael J Brzozowski and Daniel M Romero. Who should i follow? recommending people in directed social networks. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[43] Philipp M Scholl and Kristof Van Laerhoven. A feasibility study of wrist-worn accelerometer based detection of smoking habits. In *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 886–891. IEEE, 2012.

[44] Qaiser Riaz, Anna Vögele, Björn Krüger, and Andreas Weber. One small step for a man: Estimation of gender, age and height from recordings of one step by a single inertial sensor. *Sensors*, 15(12):31999–32019, 2015.

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

[46] Raphael Spreitzer. Pin skimming: Exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 51–62. ACM, 2014.

[47] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 9. ACM, 2012.

[48] Maryam Mehrnezhad, Ehsan Toreini, Siamak F Shahandashti, and Feng Hao. Touchsignatures: identification of user touch actions and pins based on mobile sensor data via javascript. *Journal of Information Security and Applications*, 26:23–38, 2016.

[49] Artur Janc and Lukasz Olejnik. Web browser history detection as a real-world privacy threat. In *European Symposium on Research in Computer Security*, pages 215–231. Springer, 2010.

[50] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A

practical attack to de-anonymize social network users. In *IEEE Symposium on Security and Privacy*, pages 223–238. IEEE, 2010.

[51] Sanae Rosen, Zhiyun Qian, and Z Morely Mao. AppProfiler: a flexible method of exposing privacy-related behavior in android applications to end users. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 221–232, 2013.

[52] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 674–689, 2014.

[53] Nan Zhang, Kan Yuan, Muhammad Naveed, Xiaoyong Zhou, and XiaoFeng Wang. Leave me alone: App-level protection against runtime information gathering on android. In *2015 IEEE Symposium on Security and Privacy*, pages 915–930. IEEE, 2015.

[54] Michael Smith, Craig Disselkoen, Shravan Narayan, Fraser Brown, and Deian Stefan. Browser history re:visited. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, Baltimore, MD, August 2018. USENIX Association.

[55] Supriyo Chakraborty, Nicolas Bitouzé, Mani Srivastava, and Lara Dolecek. Protecting data against unwanted inferences. In *2013 IEEE information theory workshop (ITW)*, pages 1–5. IEEE, 2013.

[56] Emily LC Shepard, Rory P Wilson, Lewis G Halsey, Flavio Quintana, Agustina Gómez Laich, Adrian C Gleiss, Nikolai Liebsch, Andrew E Myers, and Brad Norman. Derivation of body motion via appropriate smoothing of acceleration data. *Aquatic Biology*, 4(3):235–241, 2008.

[57] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications.* Cambridge university press, 2009.

[58] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[59] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[60] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[61] Ivan P Fellegi. On the question of statistical confidentiality. *Journal of the American Statistical Association*, 67(337):7–18, 1972.

[62] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[63] Lawrence H Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75(370):377–385, 1980.

[64] Lawrence H Cox. A constructive procedure for unbiased controlled rounding. *Journal of the American Statistical Association*, 82(398):520–524, 1987.

[65] W Fuller. Masking procedures for microdata disclosure. *Journal of Official Statistics*, 9(2):383–406, 1993.

[66] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[67] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[68] Johannes Gehrke, Michael Hay, Edward Lui, and Rafael Pass. Crowd-blending privacy. In *Annual Cryptology Conference*, pages 479–496. Springer, 2012.

[69] Michael R Siracusa, Kinh Tieu, Alexander T Ihler, John W Fisher, and Alan S Willsky. Estimating dependency and significance for high-dimensional data. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages v–1085. IEEE, 2005.

[70] AJ Paverd, Andrew Martin, and Ian Brown. Modelling and automatically analysing privacy properties for honest-but-curious adversaries. *Tech. Rep.*, 2014.

[71] Flávio du Pin Calmon and Nadia Fawaz. Privacy against statistical inference. In *50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1401–1408, 2012.

[72] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[73] Mingming Lu, Yihan Guo, Dan Meng, Cuncai Li, and Yin Zhao. An information-aware privacy-preserving accelerometer data sharing. In *International conference of pioneering computer scientists, engineers and educators*, pages 425–432, Singapore, 2017. Springer.

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

[75] David Chu, Nicholas D Lane, Ted Tsung-Te Lai, Cong Pang, Xiangying Meng, Qing Guo, Fan Li, and Feng Zhao. Balancing energy, latency and accuracy for mobile sensor data classification. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 54–67, 2011.

[76] Yunji Liang, Xingshe Zhou, Zhiwen Yu, and Bin Guo. Energy-efficient motion related activity recognition on mobile devices for pervasive healthcare. *Mobile Networks and Applications*, 19(3):303–317, 2014.

[77] Valentin Radu, Panagiota Katsikouli, Rik Sarkar, and Mahesh K Marina. A semi-supervised learning approach for robust indoor-outdoor detection with smartphones. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pages 280–294, 2014.

[78] Prafulla Kumar Choubey, Shubham Pateria, Aseem Saxena, Vaisakh Punnekkattu Chirayil SB, Krishna Kishor Jha, and Sharana Basaiah PM. Power efficient, bandwidth optimized and fault tolerant sensor management for iot in smart home. In *2015 IEEE International Advance Computing Conference (IACC)*, pages 366–370. IEEE, 2015.

[79] Akhil Mathur, Anton Isopoussu, Nadia Berthouze, Nicholas D Lane, and Fahim Kawsar. Unsupervised domain adaptation for robust sensory systems. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pages 505–509, 2019.

[80] Michaela Götz, Suman Nath, and Johannes Gehrke. MaskIt: Privately releasing user context streams for personalized mobile applications. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 289–300. ACM, 2012.

[81] Supriyo Chakraborty, Kasturi Rangan Raghavan, Matthew P. Johnson, and Mani B. Srivastava. A framework for context-aware privacy of sensor data on mobile systems. In *Proceedings of the 14th Workshop on HotMobile*, HotMobile '13, pages 11:1–11:6, New York, NY, USA, 2013. ACM.

[82] Murat A Erdogdu, Nadia Fawaz, and Andrea Montanari. Privacy-utility tradeoff for time-series with application to smart-meter data. In *AAAI 2015 Workshop on Computational Sustainability*, 2015.

[83] Tianqing Zhu, Ping Xiong, Gang Li, and Wanlei Zhou. Correlated differential privacy: hiding information in non-iid data set. *IEEE Transactions on Information Forensics and Security*, 10(2):229–242, 2015.

[84] Nazir Saleheen, Supriyo Chakraborty, Nasir Ali, Md Mahbubur Rahman, Syed Monowar Hossain, Rummana Bari, Eugene Buder, Mani Srivastava, and Santosh Kumar. mSieve: differential behavioral privacy in time series of mobile sensor data. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 706–717, 2016.

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

[86] Harrison Edwards and Amos J. Storkey. Censoring representations with an adversary. In *4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico*, 2016.

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

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

[89] Ardhendu Tripathy, Ye Wang, and Prakash Ishwar. Privacy-preserving adversarial networks. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 495–505. IEEE, 2019.

[90] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan

McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17*, pages 1175–1191, 2017.

[91] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[92] Vasyl Pihur, Aleksandra Korolova, Frederick Liu, Subhash Sankuratripati, Moti Yung, Dachuan Huang, and Ruogu Zeng. Differentially-private" draw and discard" machine learning. *arXiv preprint arXiv:1807.04369*, 2018.

[93] Aristide Charles Yedia Tossou and Christos Dimitrakakis. Achieving privacy in the adversarial multi-armed bandit. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[94] Pratik Gajane, Tanguy Urvoy, and Emilie Kaufmann. Corrupt Bandits for Preserving Local Privacy. In *Algorithmic Learning Theory*, pages 387–412, 2018.

[95] MotionSense. Motionsense dataset for human activity and attribute recognition. `https://github.com/mmalekzadeh/motion-sense`, 2018. Accessed: 01-08-2020.

[96] Piero Zappi, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Troster. Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. *Lecture Notes in Computer Science*, 4913:17, 2008.

[97] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R Millán, and Daniel Roggen. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013.

[98] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):33:1–33:33, 2014.

[99] Muhammad Shoaib, Stephan Bosch, Ozlem Incel, Hans Scholten, and Paul Havinga. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, 16(4):426, 2016.

[100] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*, pages 437–442, Belgium, April 2013.

[101] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 59:235–244, 2016.

[102] Francisco Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[103] Andrey Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922, 2018.

[104] Dawud Gordon, Jurgen Czerny, Takashi Miyaki, and Michael Beigl. Energy-efficient activity recognition using prediction. In *2012 16th International Symposium on Wearable Computers*, pages 29–36. IEEE, 2012.

[105] Xiaojun Zhu, Qun Li, and Guihai Chen. Apt: Accurate outdoor pedestrian tracking with smartphones. In *2013 Proceedings IEEE INFOCOM*, pages 2508–2516. IEEE, 2013.

[106] Hassan Ghasemzadeh, Navid Amini, Ramyar Saeedi, and Majid Sarrafzadeh. Power-aware computing in wearable sensor networks: An optimal feature selection. *IEEE Transactions on Mobile Computing*, 14(4):800–812, 2014.

[107] Sara Saeedi and Naser El-Sheimy. Activity recognition using fusion of low-cost sensors on a smartphone for mobile navigation application. *Micromachines*, 6(8):1100–1134, 2015.

[108] Xiaodong Yang, Yiqiang Chen, Hanchao Yu, Yingwei Zhang, Wang Lu, and Ruizhe Sun. Instance-wise dynamic sensor selection for human activity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1104–1111, 2020.

[109] Xin Qi, Matthew Keally, Gang Zhou, Yantao Li, and Zhen Ren. Adasense: Adapting sampling rates for activity recognition in body sensor networks. In *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 163–172. IEEE, 2013.

[110] Aftab Khan, Nils Hammerla, Sebastian Mellor, and Thomas Plötz. Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognition Letters*, 73:33–40, 2016.

[111] Weihao Cheng, Sarah Erfani, Rui Zhang, and Ramamohanarao Kotagiri. Learning datum-wise sampling frequency for energy-efficient human activity recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[112] Emily Walton, Christy Casey, Jurgen Mitsch, Jorge A Vázquez-Diosdado, Juan Yan, Tania Dottorini, Keith A Ellis, Anthony Winterlich, and Jasmeet Kaler. Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour. *Royal Society open science*, 5(2):171442, 2018.

[113] JL Hounslow, LR Brewster, KO Lear, TL Guttridge, R Daly, NM Whitney, and AC Gleiss. Assessing the effects of sampling frequency on behavioural classification of accelerometer data. *Journal of experimental marine biology and ecology*, 512:22–30, 2019.

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

[115] Tor Lattimore and Csaba Szepesvári. Bandit algorithms. *preprint*, page 28, 2018.

[116] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali

Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[117] Yu Zhao, Rennong Yang, Guillaume Chevalier, Ximeng Xu, and Zhenxing Zhang. Deep residual bidir-lstm for human activity recognition using wearable sensors. *Mathematical Problems in Engineering*, 2018, 2018.

[118] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*, pages 351–360, 2017.

[119] Jeya Vikranth Jeyakumar, Liangzhen Lai, Naveen Suda, and Mani Srivastava. Sensehar: a robust virtual activity sensor for smartphones and wearables. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 15–28, 2019.

[120] Theophano Mitsa. *Temporal data mining.* CRC Press, 2010.

[121] Wikipedia. Pearson correlation coefficient. `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient`, 2020. Accessed: 01-08-2020.

[122] World Wide Web Consortium. Accelerometer specification. `https://www.w3.org/TR/accelerometer/`, 2020. Accessed: 01-09-2020.

[123] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

[124] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[125] Abdul J Jerri. The Shannon sampling theorem–its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, 1977.

[126] Gregory W Corder and Dale I Foreman. *Nonparametric statistics: A step-by-step approach.* John Wiley & Sons, 2014.

[127] Stephen J Preece, John Y Goulermas, Laurence PJ Kenney, Dave Howard, Kenneth Meijer, and Robin Crompton. Activity identification using body-mounted sensors–a review of classification techniques. *Physiological measurement*, 30(4):R1, 2009.

[128] Yong Zhang, Yu Zhang, Zhao Zhang, Jie Bao, and Yunpeng Song. Human activity recognition based on motion sensor using u-net. *IEEE Access*, 7:75213–75226, 2019.

[129] Gaojing Wang, Qingquan Li, Lei Wang, Wei Wang, Mengqi Wu, and Tao Liu. Impact of sliding window length in indoor human motion modes and pose pattern recognition based on smartphone sensors. *Sensors*, 18(6):1965, 2018.

[130] Yi He and Ye Li. Physical activity recognition utilizing the built-in kinematic sensors of a smartphone. *International Journal of Distributed Sensor Networks*, 9(4):481580, 2013.

[131] Jamey Graham and David G Stork. Content based web advertising, October 12 2004. US Patent 6,804,659.

[132] Chromium. Site engagement. `https://www.chromium.org/developers/design-documents/site-engagement`, 2020. Accessed: 01-06-2020.

[133] Chromium. Chromium histogram guidelines. `https://chromium.googlesource.com/chromium/src.git/+/HEAD/tools/metrics/histograms/README.md`, 2020. Accessed: 01-06-2020.

[134] Royi Ronen, Elad Yom-Tov, and Gal Lavee. Recommendations meet web browsing: Enhancing collaborative filtering using internet browsing logs. *2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016*, pages 1230–1238, 2016.

[135] Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

[136] Cynthia Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19. Springer, 2008.

[137] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[138] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 729–745, 2017.

[139] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[140] Andrea Bittau, Ulfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459. ACM, 2017.

[141] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580, 2017.

[142] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1655–1658, 2018.

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

[144] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage,

and Jason. Roselander. Towards federated learning at scale: System design. In *Proceedings of the 2nd SysML Conference, Palo Alto, CA, USA*, 2019.

[145] Daniel Kifer, Solomon Messing, Aaron Roth, Abhradeep Thakurta, and Danfeng Zhang. Guidelines for implementing and auditing differentially private systems. *arXiv preprint arXiv:2002.04049*, 2020.

[146] Ninghui Li, Wahbeh H Qardaji, and Dong Su. Provably private data anonymization: Or, k-anonymity meets differential privacy. *CoRR, abs/1101.2604*, 49:55, 2011.

[147] Ali Makhdoumi, Salman Salamatian, Nadia Fawaz, and Muriel Médard. From the information bottleneck to the privacy funnel. In *2014 IEEE Information Theory Workshop (ITW 2014)*, pages 501–505. IEEE, 2014.

[148] Weina Wang, Lei Ying, and Junshan Zhang. On the relation between identifiability, differential privacy, and mutual-information privacy. *IEEE Transactions on Information Theory*, 62(9):5018–5029, 2016.

[149] Soumia Menasria, Jianxin Wang, and Mingming Lu. The purpose driven privacy preservation for accelerometer-based activity recognition. *World Wide Web*, 21(6):1773–1785, November 2018.

[150] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[151] Mario Diaz, Hao Wang, Flavio P Calmon, and Lalitha Sankar. On the robustness of information-theoretic privacy measures and mechanisms. *arXiv preprint arXiv:1811.06057*, 2018.

[152] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. volume 97 of *Proceedings of Machine Learning Research*, pages 5171–5180, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[153] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 875–884. PMLR, 2020.

[154] Liyue Fan, Li Xiong, and Vaidy Sunderam. Differentially private multidimensional time series release for traffic monitoring. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 33–48. Springer, 2013.

[155] Apple Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8), 2017.

[156] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. Local differential privacy for evolving data. In *Advances in Neural Information Processing Systems*, pages 2375–2384, 2018.

[157] Wenqi Li, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M Jorge Cardoso, et al. Privacy-preserving federated brain tumour segmentation. In *International Workshop on Machine Learning in Medical Imaging*, pages 133–141. Springer, 2019.

[158] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang

Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *arXiv preprint arXiv:2001.03618*, 2020.

[159] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019.

[160] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Minos Katevas, Hamed Haddadi, and Hamid RR Rabiee. Deep private-feature extraction. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[161] Nisarg Raval, Ashwin Machanavajjhala, and Jerry Pan. Olympus: Sensor privacy through utility aware obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2019(1):5–25, 2019.

[162] Yousef Amar, Hamed Haddadi, and Richard Mortier. An information-theoretic approach to time-series data privacy. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, page 3. ACM, 2018.

[163] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Deeprotect: Enabling inference-based access control on mobile sensing applications. *arXiv preprint arXiv:1702.06159*, 2017.

[164] Yuksel Ozan Basciftci, Ye Wang, and Prakash Ishwar. On privacy-utility tradeoffs for constrained data release mechanisms. In *2016 Information Theory and Applications Workshop (ITA)*, pages 1–6. IEEE, 2016.

[165] Yang-Sae Moon, Hea-Suk Kim, Sang-Pil Kim, and Elisa Bertino. Publishing time-series data under preservation of privacy and distance orders. In *International Conference on Database and Expert Systems Applications*, pages 17–31. Springer, 2010.

[166] Jure Sokolic, Qiang Qiu, Miguel RD Rodrigues, and Guillermo Sapiro. Learning to succeed while teaching to fail: Privacy in closed machine learning systems. *arXiv preprint arXiv:1705.08197*, 2017.

[167] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Knowledge and Information Systems*, 7(4):387–414, 2005.

[168] Gaofeng Zhang, Xiao Liu, and Yun Yang. Time-series pattern based effective noise generation for privacy protection on cloud. *IEEE Transactions on Computers*, 64(5):1456–1469, 2015.

[169] P Kingma Diederik, Max Welling, et al. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[170] Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. Differentially private mixture of generative neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1109–1121, June 2018.

[171] Ali Shahin Shamsabadi, Hamed Haddadi, and Andrea Cavallaro. Distributed

one-class learning. In *25th IEEE International Conference on Image Processing (ICIP)*, pages 4123–4127. IEEE, October 2018.

[172] Menghan Liu, Haotian Jiang, Jia Chen, Alaa Badokhon, Xuetao Wei, and Ming-Chun Huang. A collaborative privacy-preserving deep learning system in distributed mobile environment. In *CSCI*, pages 192–197. IEEE, 2016.

[173] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[174] Roshan Shariff and Or Sheffet. Differentially private contextual linear bandits. In *Advances in Neural Information Processing Systems*, pages 4296–4306, 2018.

[175] Charikleia Chatzaki, Matthew Pediaditis, George Vavoulas, and Manolis Tsiknakis. Human daily activity and fall recognition using a smartphone's acceleration sensor. In *International Conference on Information and Communication Technologies for Ageing Well and e-Health*, pages 100–118. Springer, 2017.

[176] Kleomenis Katevas, Hamed Haddadi, and Laurissa Tokarchuk. 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*, pages 375–378. ACM, 2014.

[177] George Vavoulas, Charikleia Chatzaki, Thodoris Malliotakis, Matthew Pediaditis, and Manolis Tsiknakis. The mobiact dataset: Recognition of activities of daily living using smartphones. In *ICT4AgeingWell*, pages 143–151, 2016.

[178] Supriyo Chakraborty, Kasturi Rangan Raghavan, Mani B Srivastava, Chatschik Bisdikian, and Lance M Kaplan. Balancing value and risk in information sharing through obfuscation. In *IEEE FUSION, 15th International Conference on*, pages 1615–1622, 2012.

[179] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[180] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[181] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[182] Kleomenis Katevas, Ilias Leontiadis, Martin Pielot, and Joan Serrà. Practical processing of mobile sensor data for continual deep learning predictions. In *Proceedings of the 1st International Workshop on Deep Learning for mobile systems and applications*, pages 19–24, 2017.

[183] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.

[184] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful represen-

tations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

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

[186] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

[187] Nathalie Japkowicz, Stephen Jose Hanson, and Mark A Gluck. Nonlinear autoassociation is not equivalent to pca. *Neural computation*, 12(3):531–545, 2000.

[188] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th ICDM*, pages 1096–1103. ACM, 2008.

[189] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. 2000.

[190] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 971–980, 2017.

[191] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[192] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[193] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[194] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[195] Scott R Peppet. Regulating the internet of things: first steps toward managing discrimination, privacy, security and consent. *Tex. L. Rev.*, 93:85, 2014.

[196] Daniel Rodriguez-Martin, Carlos Perez-Lopez, Albert Sama, Andreu Catala, Joan Moreno Arostegui, Joan Cabestany, Berta Mestre, Sheila Alcaine, Anna Prats, Maria Cruz Crespo, and Angels Bayes. A waist-worn inertial measurement unit for long-term monitoring of parkinson's disease patients. *Sensors*, 17(4):827, 2017.

[197] UK Biobank. About uk biobank. Available at `https://www.ukbiobank.ac.uk/about-biobank-uk/`, 2020. Accessed: 01-06-2020.

[198] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[199] Borzoo Rassouli and Deniz Gündüz. Optimal utility-privacy trade-off with the total variation distance as the privacy measure. *arXiv preprint arXiv:1801.02505*, 2018.

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

[201] Sandra Servia-Rodríguez, Liang Wang, Jianxin R Zhao, Richard Mortier, and Hamed Haddadi. Privacy-preserving personal model training. In *Internet-of-Things Design and Implementation (IoTDI), 2018 IEEE/ACM Third International Conference on*, pages 153–164. IEEE, 2018.

[202] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

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

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

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

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

[207] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[208] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[209] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

[210] Arthur T Benjamin and Jennifer J Quinn. *Proofs that really count: the art of combinatorial proof.* Number 27. MAA, 2003.

[211] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[212] Claudio Gentile and Francesco Orabona. On multilabel classification and ranking with bandit feedback. *The Journal of Machine Learning Research*, 15(1):2451–2487, 2014.

[213] Cees G.M. Snoek, Marcel Worring, Jan C. Van Gemert, Jan Mark Geusebroek, and Arnold W.M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*, 2006.

[214] Ashok N. Srivastava and Brett Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *IEEE Aerospace Conference*, 2005.

[215] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th International Conference On Machine Learning, ICML*, 2009.

[216] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2018.

[217] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[218] Lukas Köping, Kimiaki Shirahama, and Marcin Grzegorzek. A general framework for sensor-based human activity recognition. *Computers in biology and medicine*, 95:248–260, 2018.

[219] Hui-Huang Hsu, Chin-Ting Chu, Yinghui Zhou, and Zixue Cheng. Two-phase activity recognition with smartphone sensors. In *2015 18th International Conference on Network-Based Information Systems*, pages 611–615. IEEE, 2015.

[220] Sebastien Richoz, Lin Wang, Philip Birch, and Daniel Roggen. Transportation mode recognition fusing wearable motion, sound and vision sensors. *IEEE Sensors Journal*, 2020.

[221] Zhixian Yan, Vigneshwaran Subbaraju, Dipanjan Chakraborty, Archan Misra, and Karl Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *2012 16th international symposium on wearable computers*, pages 17–24. Ieee, 2012.

[222] Jung Ae Lee and Jeff Gill. Missing value imputation for physical activity data measured by accelerometer. *Statistical methods in medical research*, 27(2):490–506, 2018.

[223] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck. Adversarial preprocessing: Understanding and preventing image-scaling attacks in machine learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1363–1380. USENIX Association, 2020.

[224] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *Proceedings of the 2nd International Conference on Learning Representations, ICLR*, 2014.

[225] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[226] Mubarak G Abdu-Aguye and Walid Gomaa. Versatl: versatile transfer learning for imu-based activity recognition using convolutional neural networks. In *The 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2019.

[227] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[228] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[229] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.

[230] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[231] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[232] Jun-Ho Choi and Jong-Seok Lee. Embracenet: A robust deep learning architecture for multimodal classification. *Information Fusion*, 51:259–270, 2019.

[233] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[234] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[235] Diederik P Kingma and Ba J Adam. A method for stochastic optimization. arxiv e-prints, 2014. In *The 3rd International Conference for Learning Representations, San Diego*, 2015.

[236] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.