

Privacy-Preserving Personal Model Training

Sandra Servia-Rodríguez*, Liang Wang*, Jianxin R. Zhao*, Richard Mortier* and Hamed Haddadi†

*Department of Computer Science and Technology, University of Cambridge

†Dyson School of Design Engineering, Imperial College London

Abstract—Many current Internet services rely on inferences from models trained on user data. Commonly, both the training and inference tasks are carried out using cloud resources fed by personal data collected at scale from users. Holding and using such large collections of personal data in the cloud creates privacy risks to the data subjects, but is currently required for users to benefit from such services. We explore how to provide for model training and inference in a system where computation is pushed to the data in preference to moving data to the cloud, obviating many current privacy risks. Specifically, we take an initial model learnt from a small set of users and retrain it locally using data from a single user. We evaluate on two tasks: one supervised learning task, using a neural network to recognise users’ current activity from accelerometer traces; and one unsupervised learning task, identifying topics in a large set of documents. In both cases the accuracy is improved. We also analyse the robustness of our approach against adversarial attacks, as well as its feasibility by presenting a performance evaluation on a representative resource-constrained device (a Raspberry Pi).

Index Terms—Privacy, Machine Learning, Algorithms

I. INTRODUCTION

Large-scale data collection from individuals is at the heart of many current Internet business models. Access to these data allow companies to train models from which to infer user behaviour and preferences, typically leveraging the generous computation resources available in the public cloud. Unfortunately, this data collection is increasingly pervasive and invasive, notwithstanding regulatory frameworks such as the EU’s General Data Protection Regulation (GDPR) which attempt to restrain it. The result is that user privacy is compromised, and this is becoming an increasing concern due to reporting of the ongoing stream of security breaches that result in malicious parties accessing such personal data.

Such data collection causes privacy to be compromised even without security being breached though. For example, consider wearable devices that report data they collect from in-built sensors, e.g., accelerometer traces and heart rate data, to the device manufacturer. The device might anonymise such data for the manufacturer to use in improving their models for recognising the user’s current activity, an entirely legitimate and non-invasive practice. However, the manufacturer might fail to effectively anonymise these data and instead use them for other purposes such as determining mood, or even to sell to third-parties without the users’ knowledge. It is not only data from wearables that creates such risks: web queries, article

reads and searches, visits to shopping sites and browsing online catalogues are also indexed, analysed, and traded by thousands of tracking services in order to build preference models [1].

So far, users’ personal data were mainly sensed through their computers, their smartphones or wearable devices such as smartwatches and smart wristbands. But nowadays smart technology is entering into our homes. We are heading towards an ecosystem to the revolution where sooner or later, every device in our home will talk to an *Amazon Echo* [2], *Google Home* [3], or *Apple HomeKit* [4]. Apart from controlling smart home appliances such as light bulbs and thermostats with our voice, these smart controllers for the entire home will be required to perform more complex tasks such as detecting how many people are in the house and who they are, recognising the activity they are performing or even telling us what to wear. In this new scenario, users are becoming progressively more aware of the privacy risks of sharing their voice, video or any other data sensed in their homes with the service providers, at the same time that these applications are demanding more accurate and personalised solutions. Sending personal data to the public cloud to perform these tasks seems no longer to be an acceptable solution, but solutions should take advantage of the resource capabilities of personal devices and bring the processing locally, where data resides.

Approaches such as homomorphic encryption allow user data to be encrypted, protecting against unintended release of such data, while still being amenable to data processing. This affords users better privacy – their data cannot be used arbitrarily – while allowing data processors to collect and use such data in cloud computing environments. However, current practical techniques limit the forms of computation that can be supported. We are interested in an alternative approach where we reduce or remove the flow of user data to the cloud completely, instead moving computation to where the data already resides under the user’s control [5], [6]. This can mitigate risks of breach and misuse of data by simply avoiding it being collected at scale in the first place: attack incentives are reduced as the attacker must gain access to millions of devices to capture data for millions of users, rather than accessing a single cloud service. However, it presents challenges for the sorts of model learning processes required: how can such models be learnt without access to the users’ personal data?

In this paper we address these challenges using the *Edge Computing* paradigm. Specifically, our contributions include: (i) we develop our *personal training* method for implementing

This research was supported by the EPSRC through Grant Databox (EP/N028260/1). The work was completed while Hamed Haddadi and Sandra Servia-Rodríguez were at Queen Mary University of London.

machine learning in an environment where personal data largely remains on constrained devices under the control of the data subject (§II); (ii) we apply this method to two well-known learning tasks, one supervised (activity recognition from accelerometer traces, §III), and one unsupervised (modelling topics in text documents, §IV) and report the results; and (iii) we explore the robustness of our method against adversarial attacks, as well as the feasibility of implementing such techniques on a representative resource-constrained personal device: a Raspberry Pi 3 Model B [7] (§V).

The essence of our approach is a two-step process: (i) we first train a *shared model* using a small set of voluntarily shared users’ data and distribute this model to all users; and (ii) we then retrain this model locally using personal data held by each user, drawing inferences from the resulting *personal model*. We evaluate this approach using (i) a neural network to recognise users’ activity on the *WISDM* dataset [8] and (ii) the Latent Dirichlet Algorithm (LDA) [9] to identify topics in the Wikipedia and NIPS datasets [10], [11]. In both cases we show that the model resulting from local re-training of an initial model learnt from a small set of users performs with higher accuracy than either the initial model alone or a model trained using only data from the specific user of interest.

We also demonstrate the feasibility of training and testing a small classifier in a resource-constraint, light-weight personal device: a Raspberry Pi 3 Model B [7]. We find that such a device is certainly capable of supporting these algorithms, with negligible time to obtain inferences (on the order of milliseconds), and reasonable training times as well (on the order of tens of seconds).

II. METHODOLOGY

The current approach, which we wish to avoid, of sending all users’ personal data to the cloud for processing, is one extreme of a spectrum whose other extreme would be to train a model for a specific user using only that user’s data. For some applications, e.g., activity recognition, it has been shown that a model trained solely using data from the individual concerned provides more accurate predictions for that individual than a model trained using data from other individuals [12]. At the same time, this solution offers more privacy to the user as all computation, for both training and inference, can be done locally on the device [5]. However, this approach leads to substantial interactional overheads as training the model will likely require each user to label a significant amount of data by hand before they can obtain accurate inferences.

We propose and evaluate an alternative, hybrid approach that splits computation between the cloud and the users’ personal devices. We start by first training a model in the cloud using data from a small (relative to the population) set of users. We then distribute this *shared model* to users’ personal devices, where it can be used locally to generate inferences. In addition, it can be retrained using locally-stored personal data to become a *personal model*, specialised for the user in question.

We now describe this approach following the overview depicted in Figure 1. For clarity of exposition, we first describe

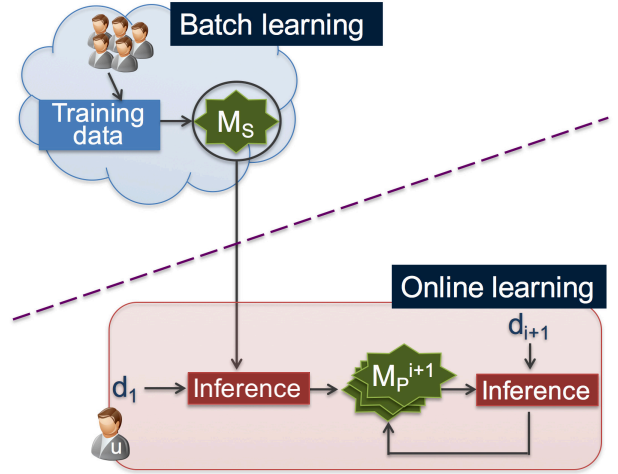


Fig. 1. Our privacy-preserving methodology for activity recognition.

our approach in the case of supervised learning, taking the activity recognition task we later use in our evaluation as a running example. We then generalise this description to other applications, including our second evaluation example of identifying topics in documents which uses an unsupervised algorithm. This suggests that any learning task, supervised or unsupervised, is amenable to our approach allowing features extracted from users’ personal data that they do not wish to disclose to be used to further personalise the initial shared model.

We start by training a *shared model*, M_S , to recognise the activity that the user is performing using data sensed with his smartphone’s built-in sensors. This *batch learning* is done on a remote server in the cloud using available public data, d_p . In the event of not having sufficient public data available for this task, data can be previously gathered from a set of users that have agreed to share their personal data perhaps by providing them with suitable incentives. To assure the confidentiality of their data as well as their presence in the dataset, the *shared model* might be obtained using differentially private training [13], [14], [15], [16].

The user u then obtains the *shared model* from the remote server. With every new sample or group of samples gathered from the smartphone’s sensors, the activity that the user is performing is locally inferred using this model. In order to allow for more accurate inferences, the user is prompted to “validate” the results by reporting the activity they were performing. The new labelled data so gathered are then used for locally retraining the model, resulting in a new *personal model*, M_P .

A. Architecture

Having described our approach, we now sketch a system architecture that might be used to implement it. This is divided into two parts: (i) residing in the cloud, the first part is responsible for constructing a *shared model* using batch

learning; and (ii) residing on each individual user’s device, the second part tunes the model from the first part using the locally available data, resulting in a *personal* model.

We identify five components in this architecture:

- 1) The **batch training module** resides in the cloud, and is responsible for training a *shared* model as the starting point using public, or private but shared, datasets that it also maintains. As this component may need to support multiple applications, it will provide a collection of different machine learning algorithms to build various needed models. It may also need to perform more traditional, large scale processing, but can easily be built using modern data processing frameworks designed for datacenters such as Mllib [17] or GraphLab [18].
- 2) The **distribution module** resides on users’ devices and is responsible for obtaining the *shared* model and maintaining it locally. In the case of very large scale deployments, standard content distribution or even peer-to-peer techniques could be used to alleviate load on the cloud service.
- 3) The **personalisation module** builds a *personal model* by refining the model parameters of the shared model using the personal data available on the user’s device. This module will also require a repository of different learning algorithms, but the nature of personal computational devices means that there will be greater resource constraints applied to the performance and efficiency of the algorithm implementation.
- 4) The **communication module** handles all the communications between peers or those between an individual node and the server. Nodes can register themselves with the server, on top of which we can implement more sophisticated membership management.
- 5) The **inference module** provides a service at the client to respond to model queries, using the most refined model available.

In our implementation, we rely on several existing software libraries to provide the more mundane of these functions, e.g., ZeroMQ [19] satisfies most of the requirements of the communication and model distribution modules, and so we do not discuss these further here.

There are many toolkits, e.g., *theano* [20] and *scikit-learn* [21], that provide a rich set of machine learning algorithms for use in the batch training and personalisation modules. However, in the case of the latter, we must balance convenience with performance considerations due to the resource-constrained nature of these devices. In light of this, we use a more recent library, Owl [22], [23], to generate more compact and efficient native code on a range of platforms, and the source code can be obtained from its Github repository¹.

B. Typical Workflow

We briefly summarise the workflow we envisage using activity recognition as an example.

- 1) When the user activates the device for the first time, the device contacts the server and registers itself in order to join the system. The device notices there is no local data for building the model, and sends a request to the server to obtain the *shared* model.
- 2) After processing the registration, the server receives the download request. The *shared* model has been trained using a initial dataset collected in a suitably ethical and trustworthy way, e.g., with informed consent, appropriate compensation, and properly anonymised. The server can either approve the download request, or return a list of peers from whom the requesting user can retrieve the model.
- 3) Having obtained the *shared* model, the device can start processing inference requests. At the same time, the device continuously collects user’s personal data, in this case, their accelerometer traces. Once enough local data is collected, the personalisation phase starts, refining the shared model to create a *personal* model.
- 4) After the *personal* model has been built, the system uses it to serve requests, and continues to refine it as more personal data is collected.

This methodology and hypothesised architecture can be applied to supervised and unsupervised learning tasks in different domains. We next show how it applies to (supervised) activity recognition (§III) and (unsupervised) topic modelling (§IV) using two well-known learning algorithms respectively.

III. ACTIVITY RECOGNITION USING ACCELEROMETER TRACES

In this section, we validate our methodology for supervised learning using a neural network to recognise users’ activity using accelerometer traces. Our evaluation for unsupervised learning is detailed in §IV, where we take the task of identifying topics in documents as a case study.

Here we consider a scenario where smartphone users want to train a motion-based activity classifier without revealing their data to others. To test the algorithms, we use the *WISDM* Human Activity Recognition dataset [8], which is a collection of accelerometer data on an Android phone by 35 subjects performing 6 activities (*walking, jogging, walking upstairs, walking downstairs, sitting* and *standing*). These subjects carried an Android phone in their front pants leg pocket while were asked to perform each one of these activities for specific periods of time. Various time domain variables were extracted from the signal, and we consider the statistical measures obtained for every 10 seconds of accelerometer samples in [8] as the $d = 43$ dimensional features in our models. Our final sample contains 5,418 accelerometer traces from 35 users, with on average 150.50 traces per user and standard deviation of 44.73.

For the purpose of validation, we compare the performance of the following models:

- *Shared*: classifier trained using data from $N - 1$ subjects and tested using data from the remaining subject;

¹<https://github.com/ryanrhymes/owl>

- *Local*: classifier trained using only data from 1 subject and tested using also data from the same subject; and
- *Personal*: classifier trained using data from $N-1$ subjects (*shared* model), retrained using data from 1 subject (*local* model) and tested using also data from the latter subject (*personal* model).

Thus, we simulated a case where a *shared* model M_S is trained with data from 34 subjects, while the *personal* model M_L is trained with different samples of data from the remaining participant (u). We then compare the accuracy of the *personal* model with the *shared* and *local* models. To this aim, we simulated two other cases: the *shared* model trained using data from 34 subjects, and the *local* model trained using only local data from u . Being S_u the samples of u and S_r the samples of all subjects but u , the samples considering for training, validation and test in each model are the following:

- *Shared*:
 - Training set: 80% of S_r
 - Validation set: 20% of S_r
 - Test set: 20% of S_u
- *Local*:
 - Training set: {1..60%} of S_u
 - Validation set: {1..20%} of S_u
 - Test set: 20% of S_u
- *Personal*: We started from the *shared* model and, in order to fine-tune it to obtain the (*personal* model), we considered the same setup as in the *local* model.

A. Multi-Layer Perceptron

We used a Multi-Layer Perceptron as the supervised learning algorithm for recognising activity using accelerometer traces. A Multi-Layer Perceptron or MLP is a type of feed-forward Artificial Neural Network that consists of two layers, input and output, and one or more hidden layers between these two layers. The input layer is passive and merely receives the data, while both hidden and output layers actively process the data. The output layer also produces the results. Figure 2 shows a graphical representation of a MLP with a single hidden layer. Each node in a layer is connected to all the nodes in the previous layer. Training this structure is equivalent to finding proper weights and bias for all the connections between consecutive layers such that a desired output is generated for a corresponding input.

The standard back-propagation learning algorithm is used for training the MLP neural architecture. For each accelerometer trace in the training set, weights and bias are modified by computing the discrepancy between the desired and actual outputs and feeding back this error to the inputs, updating the weights and bias in proportion to their responsibility for the output error. The main steps of the back-propagation algorithm are the following:

- 1) **Initialise the parameters.** All w_{ij} 's (w_{jk} 's) are initialised to small random values such as the variance of neurons in the network should be $2.0/N$ ($2.0/M$), being w_{ij} (w_{jk}) the value of the connection weight between

unit j (k) and unit i (j) in the previous layer, and N (M) the number of input (hidden) units [24]. The bias, b_j (b_k), are initialised to zero.

- 2) **Compute the class scores.** Let the individual components of an input accelerometer trace be denoted by a_i , with $i = 1, 2, \dots, N$. The output of the neurons at the hidden layer are obtained as: $H_j = \varphi(\sum_{i=1}^N a_i w_{ij} + b_j)$ with $j = 1, 2, \dots, M$, where $\varphi(\cdot)$ is the activation function and w_{ij} is the weight associated to the connection between the i -th input node and the j -th hidden node, and b_j the bias. The current recommendation is to use *ReLU* (Rectified Linear Unit) units [25], [24] as the activation function ($\varphi(x) = \max(0, x)$) though other options are possible. The outputs of the MLP are obtained using $O_k = \varphi(\sum_{j=1}^M H_j w_{jk} + b_k)$, with $k = 1, 2, \dots, C$. Here, w_{jk} is the weight associated to the connection between the j -th hidden node and the k -th output node, and b_k the bias.
- 3) **Compute the analytic gradient with back-propagation.** This is done by an iterative gradient descent procedure in the weight space which minimises the total loss between the desired and actual outputs of all nodes in the system. The *delta* terms for every node in the output layer are calculated using $\delta_k^o = (O_k - d_k)\varphi'(\cdot)$, with $k = 1, 2, \dots, C$. Here, $\varphi'(\cdot)$ is the first derivative of the activation function. Delta terms for the hidden nodes are obtained by $\delta_j^h = \sum_{k=1}^C (w_{jk} \delta_k^o)\varphi'(\cdot)$, with $j = 1, 2, \dots, M$.
- 4) **Performing a parameter update.** That is, adjust the weights and bias according to the *delta* terms and η , the learning rate parameter. For the weights, this is done using $w_{ij} = w_{ij} - \eta \delta_j^h a_i$ and $w_{jk} = w_{jk} - \eta \delta_k^o H_j$. The bias are updated according to $b_j = b_j - \eta \delta_j^h$ and $b_k = b_k - \eta \delta_k^o$.

This process is repeated until the network stabilises (converges).

In order to control the capacity of Neural Networks to prevent overfitting, ℓ_2 -regularisation is perhaps the most common form of regularisation. It can be implemented by, for every weight w in the network, adding the term $\frac{1}{2}\lambda w^2$ to the objective, where λ is the regularisation strength. Early-stopping is another mechanism to combat overfitting by monitoring the model's performance on a validation set. A validation set is a set of examples neither used for training nor for testing. During training, if the model's performance ceases to improve sufficiently on the validation set, or even degrades with further optimisation, then the training gives up on much further optimisation.

An important aspect of the MLP is the initialisation of the weights and bias, and here is also the main contribution of our proposal. For the *shared* model, M_S , we initialise the weights to random small values and the bias to zero in both layers, as suggested by previous literature [24]. However, for training the *personal* model, M_P , we start from the weights and bias of the *shared* model, M_S .

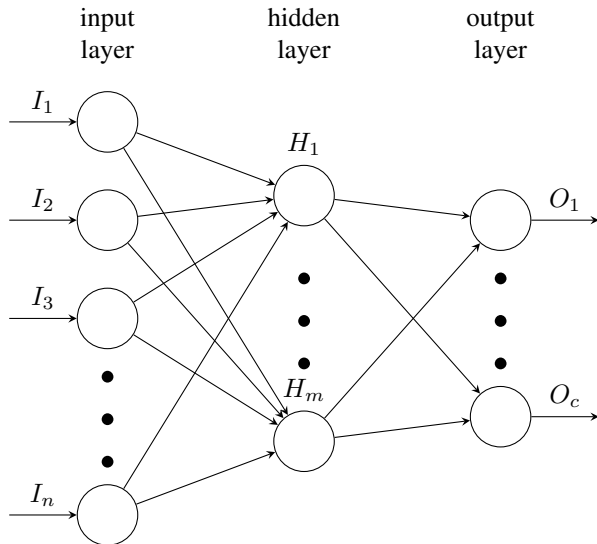


Fig. 2. The architecture of the two-layer feed-forward network.

B. Experimental setup

We set up a Multilayer Perceptron with 2 layers for activity recognition, including 1 hidden layer with 128 nodes and 1 logistic regression layer, resulting in 6,406 parameters to be determined during training. We construct the input layer using the statistical measures of users' accelerometer traces. Because of the sensitivity learning stages to feature scaling [26] we normalise all statistical measures to have zero mean and unit standard deviation. In the output layer each unit corresponds to an activity inference class, such that unit states can be interpreted as posterior probabilities.

All training procedures were implemented in python using the *Theano* deep learning library [20]. The training and testing were performed with 5-fold cross validation, using early stopping as well as ℓ_2 -regularisation to prevent overfitting. Each neuron's weight in the *shared* and *local* models was initialised randomly from $\mathcal{N}(0, 1)/\sqrt{2.0/n}$, where n is the number of its inputs, and biases were all initialised to zero. Parameters in the *personal* model were initialised to the values obtained in the *shared* model. Finally, we used grid search to determine the values of the hyper-parameters, setting the learning rate to 0.05 for the *shared* model and to 0.001 for the *local* and *personal* models, and the ℓ_2 -regularisation strength to $1e^{-5}$ for all the models. The training epochs were set to 1000 in all models, while the batch size was set equal to the size of the training sets in the *shared* model, and to 1 (online learning) in the *local* and *personal* ones. The reasons behind this are the small size of the dataset, and the availability of the training samples in a real scenario (samples for the *shared* model can be assumed to be all available for training, whereas samples in the *local* and *personal* models become available for training as time goes by).

We repeated the experiment for each participant, using 5-fold cross-validation and different number of samples to train the *local* and *personal* models. In each simulation of every

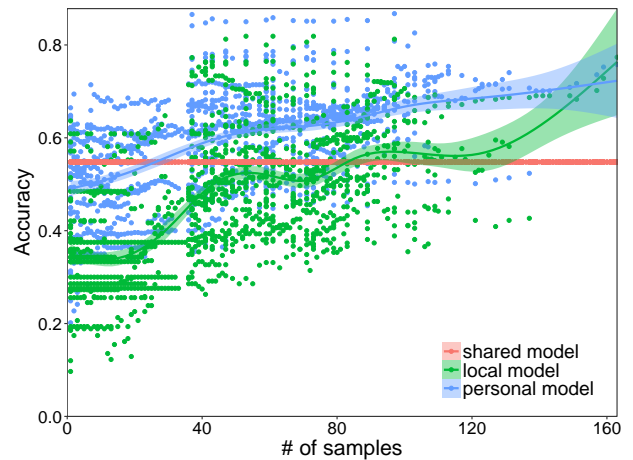


Fig. 3. Accuracy obtained with each model for different number of local samples per user.

user, we incremented in 1 the number of samples used for training, and also incremented in 1 the samples used for validation until reaching 60% of samples for training and 20% for validation, respectively.

C. Results

Figure 3 reports the accuracy achieved with each model when considering different number of local samples per user. Results show that the effect of training or retraining a model with few samples from the individual under test produces worse predictions than using samples from other individuals (*shared* model). That is, while the model is adapting to the new *scenario*, the performance of the prediction slightly drops. However, when more samples (20 on average or more) are used to retrain this *shared* model, the accuracy of the prediction exceeds the accuracy obtained with the *shared* model itself. Specifically, the accuracy increases with increments in the number of samples used for retraining the model. That is, the more local samples considered to retrain the model, the more *personalised* it becomes for the considered individual. However, although the improvement on the accuracy with the increment of the number of samples is also shared with the *local* model, more samples per individual are required for training a model from scratch (*local* model) in order to obtain the same accuracy than when starting from a *shared* model (*personal* model). We also observe that, after on average 163 samples, the *local* model performs better than the *personal* model. However, this is not significant, since there is one unique user in the dataset with that number of samples or higher available for training. In summary, (i) retraining a *shared* model locally using 20 or more samples from the user increases the accuracy with respect to that obtained with the *shared* model, and (ii) to obtain the same accuracy when training a model from scratch using only local samples, more than 150 training samples are required on average.

IV. TOPIC MODELLING OF PERSONAL TEXT CORPUS

Text corpus, such as web pages, documents, e-books and emails, is one of the most widely distributed media on the Internet and also dominates many users' devices. Text classification therefore has a wide application to facilitate people's daily life by grouping or filtering the items in a text corpus based on certain topics. Because personal text corpus often contains a significant amount of private information, uploading such corpus to a public cloud will certainly breach user privacy.

Consider now a scenario where users wish to classify the textual documents on their computers without revealing their content to others. Being more precise, we consider researchers working on a confidential project within a company that do not wish to disclose the publications they are reading, but they do wish to have the documents classified according to their content. To simulate such a scenario, we use two text datasets in our evaluation: the NIPS [10] and the Wikipedia [11] datasets. The NIPS dataset is a collection of papers published in NIPS conference over the past two decades. It contains about 1.5k papers and 1.9 million words. For Wikipedia, we download its latest English dump in January 2017 which contains about 5 million articles and 2.9 billion words. For the purpose of validation, we compare the performance of the following models:

- *Local*: topic extraction using only data from the NIPS dataset (local data);
- *Personal*: topic extraction using only data from the Wikipedia dataset (*shared* model), and retraining using data from the NIPS dataset (*local* model).

Thus, we simulated a case where a *shared* model M_S is trained with data from the Wikipedia dataset, while the *personal* model M_L is trained with the NIPS dataset. We then compare the accuracy of the *personal* model with the *local* model. To this aim, we also simulated the case where the *local* model trained using only local data from the NIPS dataset.

A. Latent Dirichlet Allocation

A **topic model** is an effective tool for text mining. It aims to construct a statistical model to represent the abstract "topics" contained in a collection of documents. By doing so, similar documents can be grouped together for future queries. As each document is composed of a sequence of words, it is often represented as a very high-dimensional and sparse vector using a bag-of-words model. Each dimension represents one unique vocabulary in the dictionary extracted from the corpus, and the magnitude of each dimension is often calculated as the term frequency in the corresponding document. Therefore, the dimensionality of these vectors depends on the size of dictionary, and it is common the dimensionality is over dozens of thousands.

LDA is a generative model which explicitly models topics as latent variables based on the co-occurrences of terms and documents in a text corpus. LDA is similar to pLSA [27] but replaces the maximum likelihood estimator with Bayesian

estimator, hence it is sometimes referred to as the Bayesian version of pLSA. LDA assumes that the topic distribution has a Dirichlet prior.

As each document can be represented as a vector, finding a given document's similar documents is equivalent to search for its k -nearest neighbours in the high-dimensional space. Many prior works [28], [29], [30], [31] focus on building compact and efficient data structure and search algorithm to speed up the queries to the models.

As mentioned, LDA is an unsupervised learning method and its goal in training is to maximise its likelihood function. Its model contains two important parameters:

- **Document-Topic distribution**: it indicates the probability distribution of each document over a set of topics.
- **Topic-Word distribution**: it indicates the probability distribution of each topic over a set of words extracted from the text corpus.

Essentially, the two parameters are represented as two matrices in the algorithm containing the information of document-topic and topic-word assignment respectively. A typical training task can be divided into two phases: First, the two parameters will be initialised by assuming both have a Dirichlet prior; second, the model will be updated by applying Collapsed Gibbs Sampling to all the documents. The second phase will be applied iteratively until we reach the pre-defined number of iterations or the model converges. To evaluate the effectiveness of an LDA model, we can measure the log likelihood of the constructed model over a test data set. The log likelihood indicates how well the model can interpret the given data set, and the higher value it is, the better it is.

B. Experiment setup

In our topic modelling scenario, a user u owns a set of text documents, D_u , and wants to identify their topics without revealing their content. There is another set of publicly available documents, D_r , that he can benefit from. The set of u 's documents, D_u , is composed by the documents in the NIPS dataset, whereas D_r is formed by different random samples of documents from the Wikipedia dataset.

We start by building a *shared* LDA model called M_S out of the public available documents D_r as we did in the previous activity recognition case. However, one thing worth noting here is that M_S only includes the Topic-Word distribution (as well as a dictionary to tokenise the documents) which is a very sparse matrix. The Document-Topic parameter depends on the specific text corpus and is not useful for others to initialise a *personal* model, hence it is not necessary to include into M_S .

We then build the new LDA model, *personal* model or M_P , using the new document samples that corresponds to users' personal data, i.e., D_u , and compare it with the *local* model trained solely using user's documents. Specifically, the method for building the *local* model is just repeating the typical process of building the model from scratch. Alternative method, i.e. the method to build the *personal* model, is to request M_S and use it to initialise the local Document-Topic distribution parameter instead of assuming a Dirichlet prior.

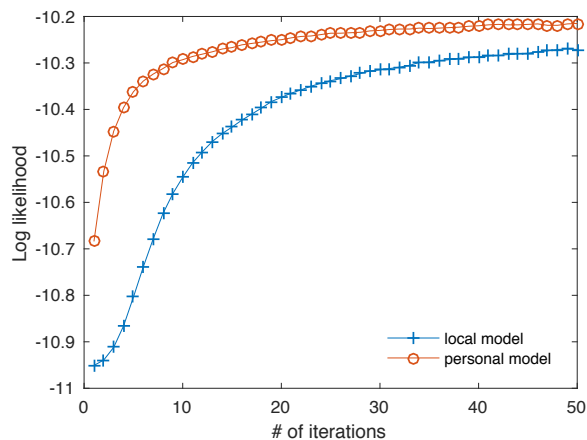


Fig. 4. Log likelihood in each iteration while building a LDA model with and without using a *shared* model M_S . That is, for building the *personal* and *local* LDA models.

In the following, we will compare and present the accuracy and efficiency of the *local* and *personal* models.

We use the topic modelling module in Owl [22] library to perform the aforementioned experiments. Each experiment is repeated 10 times to guarantee its consistency.

C. Improved accuracy and efficiency

Figure 4 presents the evolution of log likelihood in each iteration while building the *personal* (M_P) and *local* models. NIPS dataset is used in this first experiment, and the *shared* model, M_S , is trained using 50% of the data. A user’s local data are generated by randomly selecting 300 documents from the rest of the dataset.

The two lines in the figure correspond to the two ways of building the user’s topic model: the red line is for using M_S to initialise model (*personal* model) whereas the blue line is for building the model from scratch by solely using the user’s local documents (*local* model). As we see, the *personal* model is able to achieve higher likelihood in each iteration than the *local* one. Meanwhile, it also indicates that the model is able to converge much faster given a target accuracy.

The NIPS dataset is larger than the activity recognition dataset, which allows us to raise and answer the next question: *how much public data do we need to use for building the shared model M_S ?* This question becomes very relevant especially when the amount of public or shared data is limited. In the next experiment, while keeping the rest of the experiment settings the same, we build multiple *shared* models M_S by increasing the amount of documents for training step by step, from 100 to 1000. With these new M_S , we repeat the same experiment presented in Figure 4 to investigate how the amount of public data used to train M_S impacts the personal training (*personal* model). Figure 5 presents our results. The blue line with “+” marker at the bottom is the same as that in Figure 4, representing the training without using M_S (*local* model). The rest of the lines represent the log likelihood of the M_S using different percentage of the 1000 documents as

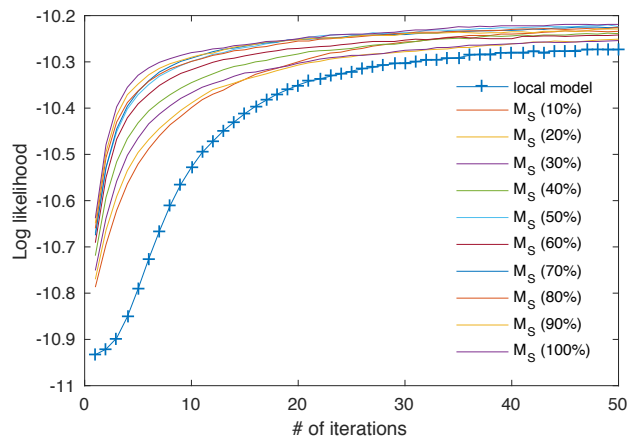


Fig. 5. Log likelihood in each iteration while building a *personal* LDA model by including different amount of public data into the *shared* model M_S .

the public sample. All M_S models are trained using a fixed number (i.e., 50) of iterations.

Results in Figure 5 show that including more data in M_S will certainly improve the efficiency and accuracy when localising or personalising the *shared* model which is reflected as gradually improved likelihood in each line. Note that including more data in training M_S will not increase the parameter size therefore it does not introduce extra overhead in distributing the *shared* model. However, these results also deliver another important message, namely such benefits drop quickly as we add more and more data. The most significant improvement appears in the very beginning when we only include a small amount of data (i.e., 10%). To some extent, it justifies our proposal by showing a small amount of public data is able to boost local training.

D. Topic-based local dataset

Wikipedia dataset contains a rich set of meta information such as manually assigned topic categories, which can help us in simulating users who have different interests in various topics. Users of different interest may possess a rather different local dataset from each other. The question is whether our method is still effective when the topics in the local data are only a subset of those included in the documents used for training the *shared* model.

To answer the question, we design other experiment wherein when we generate the local dataset, we only randomly select the articles from a pre-determined topic, e.g., computer networking, architecture design, British history. We generate multiple local datasets using different topics. The x -axis in Figure 6 shows the topic indices in the Wikipedia dataset, and we present the results of 10 of those selected topics. The way of selecting the shared data for training M_S remains the same. In total, we randomly sample 1000 articles from the whole Wikipedia as public or shared data, and 500 articles of a given topic for each local dataset.

We first measure the absolute amount of improvement in log likelihood between the first and the tenth iteration while

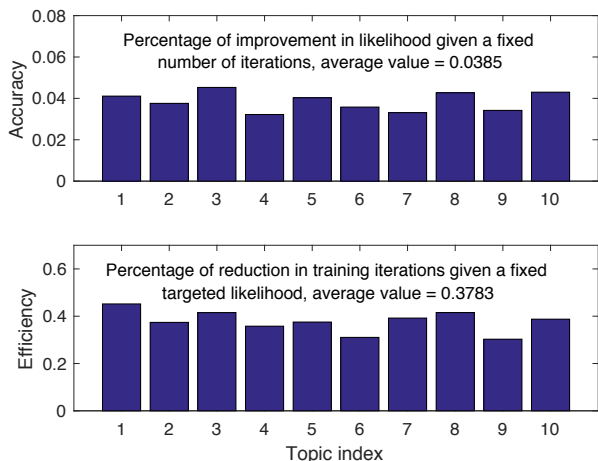


Fig. 6. *Shared* model also helps in improving accuracy and efficiency even when the local datasets have a strong focus on specific topics. Ten randomly selected topics from Wikipedia are used to generate local dataset. The upper figure presents the percentage of improving log likelihood for a fixed number of iterations, while the lower figure measures the percentage of reduction in training iterations for a targeted likelihood. Both figures have the same x-axis which is indexed by topic numbers.

training a *local* model, based on which we then calculate how much we can improve by starting with a *shared* model (*personal* model). The results in upper part of Figure 6 show that the improvement varies between 3.22% to 4.53% with an average equals to 3.85%. We also measure the improvement in efficiency by investigating how much we can reduce the number of iterations to reach a targeted log likelihood with the help of a *shared* model. In this experiment, we set the targeted log likelihood to -10.203 and the lower part of Figure 6 presents our results. Similar to the upper one, using a *shared* model can significantly boost the training efficiency, with the minimum over 30% reduction in iterations over all cases. On average, we are able to save over 37.8% iterations. Another thing worth mentioning here is that since the *shared* model is already sparse, it further reduces the time spent in each iteration than training purely on local data (*local* model) which needs to start with a highly dense local model. This indicates the saving is even more significant in terms of absolute amount of training time reduced.

V. PRACTICAL CONSIDERATIONS

In the following, we discuss the privacy guarantees of our methodology and its robustness against adversarial attacks (§V-A). We then demonstrate the feasibility of its deployment by presenting a performance evaluation on a representative resource-constrained device (§V-B).

A. Adversarial attacks

Our system can suffer the attacks and consequences of malicious users. There are several potential attacks against any learning system [32], [33]. Here we focus on how privacy and causative attacks might affect our system. On a privacy attack the adversary obtains information from the learner, compromising the secrecy or privacy of the system’s users. The

aim of a causative attack is on altering the parameters of the target model by manipulating the training dataset. An example of this type of attacks are poisoning attacks, where an attacker may poison the training data by injecting carefully designed samples to eventually compromise the whole learning process. The target model then updates itself with the poisoned data and gradually compromises. Below we describe the potential effects of these attacks in our system.

1) *Privacy attacks*: Our solution guarantees the confidentiality of users’ data (potential users) given that their devices are not compromised, since their personal data never leave their devices. Since both the data and the personal model resides on the user’s device, attacks such as model inversion [34] –where an attacker, given the model and some auxiliary information about the user, can determine some user’s raw data; and membership query [35], where, given a data record and black-box access to a model, an adversary could determine if the record was in the model’s training dataset, cannot affect our users. However, we cannot assure the confidentiality of the data, neither robustness against these attacks, for those users that have freely agreed to share their data in the same way as the big corporations are not doing so with their customers data. For many applications we envisage and describe in the introduction, such as those based on object recognition or those that work with textual data, there is already a large amount of data freely available on the Internet with which to build the shared model, and whose confidentiality does not need to be guaranteed. On the other hand, for applications such as face or speaker recognition, techniques based on differentially private training [13], [14], [15], [16] could be applied in order to, a priori, guarantee the confidentiality of the volunteers’ data. On the contrary, the training of the personal model for the final users happens locally on their devices so that neither their data nor their personal model leave their devices, and its confidentiality is guaranteed by the security offered by their device, security that is out of the scope of the methodology proposed here.

2) *Poisoning attacks*: We envisage two different points or steps in our system that adversaries might wish to attack: when building the shared model in a remote server in the public cloud using public data available or *shared* by a group of volunteers, and when personalising the model by local retraining in the user’s device (*personalisation*). In the case of a poisoning attack to our proposed methodology, the shared model can be corrupted by malicious volunteers poisoning the data with fake samples. However, during the local retraining, if the adversary wishes to corrupt the personal model, he needs to gain access to the local device of the user to poison the data and fool the model. Poisoning the data to train the personal model needs the attacker to gain access to the local device of the user.

B. Deployment Feasibility

The success of our privacy-preserving methodology for learning personal models is conditional on the ability to run model refinement in *near* real-time on resource-constrained

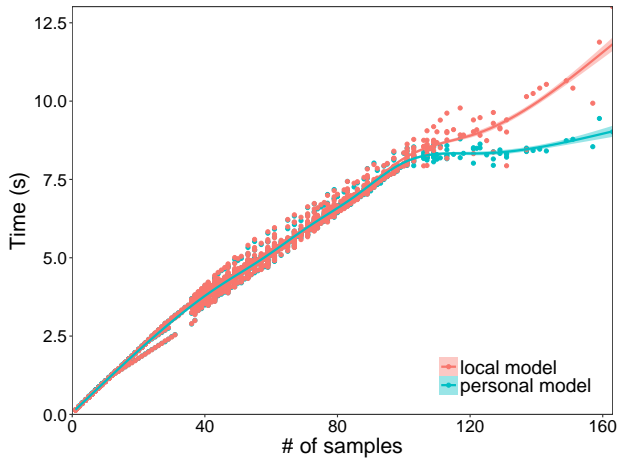


Fig. 7. Time for training with each model, varying the number of local samples per user.

personal devices which lack the capabilities of cloud-based servers. These environments are of increasing interest for deployment of such techniques as availability of computation resources outside datacenters continues to increase with creation of “fog computing” environments using cheap and energy-efficient platforms such as those based on ARM processors [36].

To verify the deployment feasibility of our approach on resource-constrained personal devices we evaluate, for both tasks, the second phase of our approach, testing the refinement and inference aspects on a Raspberry Pi 3 Model B [7] as representative of these sorts of environments.

Figure 7 shows the time taken for training a *personal* model (refining the initial *shared* model), and for the alternative approach of learning a *local* model using only a single user’s data available locally, for the setup in §III. In both cases training takes seconds to complete, with time increasing linearly with the number of samples considered due to the online nature of the training process (only one sample is considered at every update of the model). The time for making the inference is insignificant compared with the time for training, being on the order of milliseconds.

Figure 8 reproduces results from §IV on the same resource-constrained device using the NIPS dataset. Figures 8(a-b) show the time spent per iteration while building the model, respectively varying the number of training documents while keeping the vocabulary size fixed, and the vocabulary size while keeping the number of training documents fixed. Each test is repeated for 20 iterations. In both cases, we observe the expected linear growth in model training time.

Figures 8(c-d) show the time it takes to reach a certain level of likelihood (we use -10.230 as the threshold, as in the previous experiments) given different *shared* models. All models are trained with 200 documents. The results align with what we have found out in Figure 5: including more data decreases both the time and number of iterations required to reach the desired precision threshold. This decrease is most

obvious when the model moves from not using any data to including a small amount (10%). As further data are included, the benefits quickly diminish.

VI. RELATED WORK

Data-driven solutions are now pervasive in areas such as advertising, smart cities and eHealth [37], [38]. Almost everything we do in our daily lives is tracked by some means or another. Although the careful analysis of these data can be highly beneficial for us as individuals and for the society in general, this approach usually entails invasion of privacy, a high price that progressively more people are not willing to pay [39].

Several privacy-preserving analytical solutions have been proposed to guarantee the confidentiality of personal data while extracting useful information [40], [41], [42], [43]. Prominent among them are those that build on Dwork’s *differential privacy* framework [44], [45], [46], [34], [47], which formalises the idea that a *query* over a sensitive database should not reveal whether any one person is included in the dataset [48]. In the case of machine learning, the idea is that a differentially-private model should not reveal whether data from any one person were used to train the model. Most of these techniques for differentially-private machine learning are usually based on adding noise during the training, which leads to a challenging trade-off between accuracy and privacy.

Distributed machine learning (DML) arised as a solution to well utilize large computer clusters and highly parallel computational architectures to speed up the training of big models over the large amounts of data available nowadays [49]. Systems dealing with very large datasets have already had to handle the case where no single node can contain and process the entire dataset, but the dataset and/or the model to learn are parallellised among different machines, models are sequentially trained on each single machine, and some sort of synchronisation mechanism is applied to aggregate the parameters of the model to learn [50], [51], [52], [53]. DML may also be a potential solution when the volume of the data is not the main issue, but the distribution occurs when different entities own different datasets which, if aggregated, would provide useful knowledge. However, the sensitivity of such data often prevents these entities from sharing their datasets, restricting access to only a small set of selected people as in the case of patients’ medical records [43]. Several solutions have been proposed for privacy-preserving distributed learning, where information is learnt from data owned by different entities without disclosing either the data or the entities in the data. Shokri and Shmatikov [13] and McMahan *et al.* [14] propose solutions where multiple parties jointly learn a neural-network model for a given objective by sharing their learning parameters, but without sharing their input datasets. A different approach is proposed by Hamm *et al.* [15] and Papernot *et al.* [16], where privacy-preserving models are learned locally from disjoint datasets, and then combined on a privacy-preserving fashion. However, the privacy guarantees

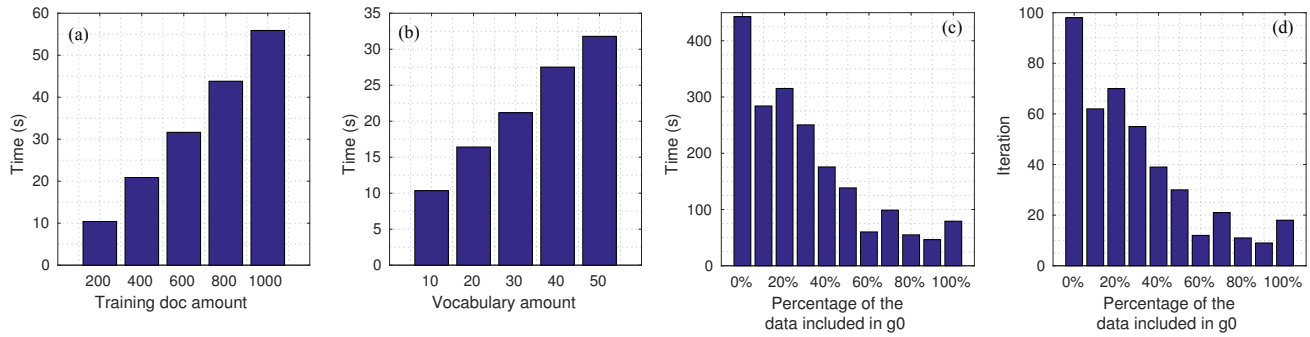


Fig. 8. Time taken to reach a given likelihood (-10.230) started from different *shared* models. The y -axis is the time in seconds, and x -axis is the percentage of the data included in the initial *shared* model.

of some of these solutions have recently been called into question [54].

Contrary to previous approaches, our aim is not on learning a global model from sensitive data from multiple parties, but to learn a personalised model for each individual party that builds on a model learnt from a relatively small set of others parties, without requiring access to their raw data. We build *personal* learning models similar to the personal recommender system by Balasubramanian *et al.* [55] but generalising the solution to any learning algorithm. Our solution takes advantage of transfer learning [56] to achieve better performance than algorithms trained using only local data, particularly in those common situations where local data is a scarce resource. This solution brings most of the data processing to where the data resides and not the other way around, exactly as the edge computing paradigm calls for [57]. Recent work have demonstrated the feasibility of running complex deep learning inferences on local devices such as smartphones [58], [59]. While in these works models are previously trained in an offline manner, our experiments in §V-B proved that both the inference and the local retraining can be performed locally on a low-power device such as the Raspberry Pi in a timely manner.

VII. CONCLUSION

Our privacy-preserving methodology for learning analytics relies on the ability of current personal devices such as smartphones, tablets and small form-factor computers such as the Raspberry Pi to carry out traditionally resource-demanding tasks. By splitting model training between the cloud and the personal device, we avoid sending personal data to untrustworthy remote entities in the cloud while maintaining efficiency and improving accuracy of both training and inference. Users thus keep all rights over their personal data while retaining the benefits of learning-based services.

We demonstrated our methodology for two typical learning tasks, one supervised and one unsupervised: activity recognition from accelerometer data, and identification of topics in text documents. Our experiments showed improvements both in accuracy and efficiency using this methodology with respect to both traditional cloud-based solutions and solutions based on training the model using *only* the data available

from the user to whom we are providing the service. We also demonstrated the feasibility of our approach by examining performance of implementation of the local step of our methodology on a Raspberry Pi 3 Model B.

Our results prove that this approach is promising, and we believe that it is widely applicable and able to positively improve the privacy-preservation of applications where data is distributed between entities whose confidentiality needs to be secured. In a world where the Internet of Things is connecting progressively more devices everyday – some of which already reside in our home, and the rising concerns about the possibility of private data leaving or getting stolen from our smart homes, our privacy-preserving methodology comes to the aid of designers and developers of smart home applications to fulfil the need of privacy guarantees that their users demand. However, there are certainly areas appropriate for future work. For example, our current solution does not leverage any cooperation between user nodes, and we plan to explore a *cooperative learning* approach to allow *similar* users to jointly train machine learning models in cases where they do not individually have sufficient labelled data locally to retrain a *shared* model, likely the case during the first moments they use the learning service. A potential solution to boost the accuracy may deal with similar users retraining the *shared* model collaboratively, in a privacy-preserving fashion.

REFERENCES

- [1] M. Falahastegar, H. Haddadi, S. Uhlig, and R. Mortier, “Tracking personal identifiers across the web,” in *Passive and Active Measurement conference (PAM 2016)*, 2016.
- [2] “Amazon echo,” <https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E/>, No date, accessed May 25, 2017.
- [3] “Google home,” <https://madeby.google.com/home/>, No date, accessed May 25, 2017.
- [4] “Apple homekit,” <https://www.apple.com/ios/home/>, No date, accessed May 25, 2017.
- [5] A. Chaudhry, J. Crowcroft, H. Howard, A. Madhavapeddy, R. Mortier, H. Haddadi, and D. McAuley, “Personal data: Thinking inside the box,” in *Proceedings of The Fifth Decennial Aarhus Conference on Critical Alternatives*, ser. AA ’15. Aarhus University Press, 2015, pp. 29–32. [Online]. Available: <http://dx.doi.org/10.7146/aahcc.v1i1.21312>
- [6] R. Mortier, J. Zhao, J. Crowcroft, L. Wang, Q. Li, H. Haddadi, Y. Amar, A. Crabtree, J. Colley, T. Lodge, T. Brown, D. McAuley, and C. Greenhalgh, “Personal data management with the Databox: What’s

- inside the box?" in *Proc. Cloud Assisted Networking workshop at ACM CoNEXT*, Dec. 12 2016.
- [7] "Raspberry pi," <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>, No date, accessed February 15, 2017.
 - [8] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.
 - [9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003. [Online]. Available: <http://dl.acm.org/citation.cfm?id=944919.944937>
 - [10] "Nips - bag of words data set," <https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words/>, No date, accessed January 20, 2017.
 - [11] "Wikipedia dataset," <https://dumps.wikimedia.org/enwiki/latest/>, No date, accessed January 20, 2017.
 - [12] G. M. Weiss and J. W. Lockhart, "The impact of personalization on smartphone-based activity recognition," in *AAAI Workshop on Activity Context Representation: Techniques and Languages*, 2012.
 - [13] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.
 - [14] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.
 - [15] J. Hamm, P. Cao, and M. Belkin, "Learning privately from multiparty data," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 555–563.
 - [16] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.
 - [17] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *Journal of Machine Learning Research*, vol. 17, no. 34, pp. 1–7, 2016.
 - [18] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," *Proc. VLDB Endow.*, vol. 5, no. 8, pp. 716–727, Apr. 2012. [Online]. Available: <https://doi.org/10.14778/2212351.2212354>
 - [19] "Zeromq - distributed messaging," <http://zeromq.org>, No date, accessed January 20, 2017.
 - [20] "Theano deep learning," <http://deeplearning.net/software/theano>, No date, accessed January 20, 2017.
 - [21] "scikit-learn," <http://scikit-learn.org/>, No date, accessed January 20, 2017.
 - [22] L. Wang, "Owl: A general-purpose numerical library in ocaml," 2017. [Online]. Available: <http://arxiv.org/abs/1707.09616>
 - [23] L. Wang, B. Catterall, and R. Mortier, "Probabilistic Synchronous Parallel," *ArXiv e-prints*, 2017.
 - [24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
 - [25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.
 - [26] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.
 - [27] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '99. New York, NY, USA: ACM, 1999, pp. 50–57. [Online]. Available: <http://doi.acm.org/10.1145/312624.312649>
 - [28] L. Wang, S. Tasoulis, T. Roos, and J. Kangasharju, "Kvasir: Scalable provision of semantically relevant web content on big data framework," *IEEE Transactions on Big Data*, vol. 2, no. 3, pp. 219–233, Sept 2016.
 - [29] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, "Fast approximate nearest-neighbor search with k-nearest neighbor graph," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI'11. AAAI Press, 2011, pp. 1312–1317. [Online]. Available: <http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-222>
 - [30] J. He, W. Liu, and S.-F. Chang, "Scalable similarity search with optimized kernel hashing," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. New York, NY, USA: ACM, 2010, pp. 1129–1138. [Online]. Available: <http://doi.acm.org/10.1145/1835804.1835946>
 - [31] V. Hyvönen, T. Pitkänen, S. Tasoulis, E. Jääsaari, R. Tuomainen, L. Wang, J. Corander, and T. Roos, "Fast nearest neighbor search through sparse random projections and voting," in *2016 IEEE International Conference on Big Data (Big Data)*, Dec 2016, pp. 881–888.
 - [32] M. Barreno, B. Nelson, A. D. Joseph, and J. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010.
 - [33] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. ACM, 2011, pp. 43–58.
 - [34] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *USENIX Security*, 2014, pp. 17–32.
 - [35] R. Shokri, M. Stronati, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proceedings of the 38th IEEE Symposium on Security and Privacy*, 2017.
 - [36] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: <http://doi.acm.org/10.1145/2342509.2342513>
 - [37] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314 – 347, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0020025514000346>
 - [38] X. Wu, X. Zhu, G. Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, Jan 2014.
 - [39] L. Brandimarte, A. Acquisti, and G. Loewenstein, "Misplaced confidences," *Social Psychological and Personality Science*, vol. 4, no. 3, pp. 340–347, 2013. [Online]. Available: <http://dx.doi.org/10.1177/1948550612455931>
 - [40] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00. New York, NY, USA: ACM, 2000, pp. 439–450. [Online]. Available: <http://doi.acm.org/10.1145/342009.335438>
 - [41] C. C. Aggarwal and S. Y. Philip, "A general survey of privacy-preserving data mining models and algorithms," in *Privacy-preserving data mining*. Springer, 2008, pp. 11–52.
 - [42] Z. Erkin, J. R. Troncoso-pastoriza, R. L. Lagendijk, and F. Perez-Gonzalez, "Privacy-preserving data aggregation in smart metering systems: an overview," *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 75–86, March 2013.
 - [43] G. Bellala and B. Huberman, "Securing private data sharing in multi-party analytics," *First Monday*, vol. 21, no. 9, 2016. [Online]. Available: <http://www.firstmonday.dk/ojs/index.php/fm/article/view/6842>
 - [44] A. D. Sarwate and K. Chaudhuri, "Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data," *IEEE signal processing magazine*, vol. 30, no. 5, pp. 86–94, 2013.
 - [45] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 245–248.
 - [46] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011.
 - [47] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.
 - [48] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.
 - [49] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Advances in Neural Information Processing Systems*, 2014, pp. 19–27.
 - [50] T. H. Cormen and M. T. Goodrich, "A bridging model for parallel computation, communication, and i/o," *ACM Computing Surveys (CSUR)*, vol. 28, no. 4es, p. 208, 1996.

- [51] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *Advances in Neural Information Processing Systems*, 2011, pp. 873–881.
- [52] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Advances in neural information processing systems*, 2013, pp. 1223–1231.
- [53] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.
- [54] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," *CoRR*, vol. abs/1702.07464, 2017. [Online]. Available: <http://arxiv.org/abs/1702.07464>
- [55] S. Jain, V. Tiwari, A. Balasubramanian, N. Balasubramanian, and S. Chakraborty, "Pria: A private intelligent assistant," in *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '17. New York, NY, USA: ACM, 2017, pp. 91–96. [Online]. Available: <http://doi.acm.org/10.1145/3032970.3032988>
- [56] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [57] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [58] P. Georgiev, N. D. Lane, K. K. Rachuri, and C. Mascolo, "Dsp. ear: Leveraging co-processor support for continuous audio sensing on smartphones," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014, pp. 295–309.
- [59] —, "Leo: Scheduling sensor inference algorithms across heterogeneous mobile processors and network resources," in *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, no. CONF CODE. ACM, 2016, pp. 320–333.