Blockchain-Based Privacy Preserving Deep Learning

Xudong Zhu[1(✉)], Hui Li[2], and Yang Yu[1]

[1] Xi'an University of Architecture Technology, Xi'an 710055, Shaanxi, China
zhudongxu@vip.sina.com
[2] Xidian University, Xi'an 710126, Shaanxi, China

Abstract. Smart mobile devices have access to huge amounts of data appropriate to deep learning models, which in turn can significantly improve the end-user experience on mobile devices. But massive data collection required for machine learning introduce obvious privacy issues. To this end, the notion of federated learning (FL) was proposed, which leaves the training data distributed on the mobile devices, and learns a shared model by aggregating locally-computed updates. However, in many applications one or more Byzantine devices may suffice to let current coordination learning mechanisms fail with unpredictable or disastrous outcomes. In this paper, we provide a proof-of-concept for managing security issues in federated learning systems via blockchain technology. Our approach uses decentralized programs executed via blockchain technology to establish secure learning coordination mechanisms and to identify and exclude Byzantine members. We studied the performance of our blockchain-based approach in a collective deep-learning scenario both in the presence and absence of Byzantine devices and compared our results to those obtained with an existing collective decision approach. The results show a clear advantage of the blockchain approach when Byzantine devices are part of the members.

Keywords: Smart mobile device · Federated learning · Byzantine devices · Blockchain technology

1 Introduction

More and more smart mobile devices, such as mobile phones and tablet computers, have become the main computing equipment for most people [1]. The powerful sensors (such as cameras, accelerometers, and GPS) on these mobile devices which are frequently carried produce and have access to an unprecedented amount of data, much of it private in nature. Models learned on such data can greatly improve usability by powering more smart applications, but the private nature of the data means there are risks and responsibilities to storing it in a centralized location.

Supported by Natural Science Fund of Shaanxi Province #K05074.

To address the challenges of the private data inclusion in deep learning, McMahan et al. [2] proposed federated learning, a learning technique that allows users to collectively reap the benefits of shared models trained from this rich data, without the need to centrally store it. Each participating device has a local training dataset which is never uploaded to the server. And each participating device computes an update to the current global model maintained by the server, and only this update is communicated. A principal advantage of this approach is the decoupling of model training from the need for direct access to the raw training data. FL can significantly reduce privacy and security risks by limiting the attack surface to only the device, rather than the device and the cloud.

But federated learning network is often claimed to be highly fault-tolerant, in some cases one or more Byzantine devices – devices that show arbitrarily faulty or malicious behavior – may suffice to let current coordination learning mechanisms fail. In real-world, devices in federated learning network will face situations in which some of the devices become Byzantine devices. Robustness to Byzantine devices will therefore become of paramount importance. Until now federated learning research has left unaddressed the problem of how to manage the security issues generated by the presence of Byzantine devices: (i) *tampered devices or failing sensors:* the messages sent from these devices can contain wrong or deceptive information; (ii) *attacked or noisy communication channels*: messages can be manipulated or destroyed while propagating through the peer-to-peer network; (iii) *loss of availability*: information stored on a device's hard drive might be deleted; the device might be captured or destroyed.

In this paper, we argue that blockchain technology might be used to provide solutions to the aforementioned security issues. In particular, we show that it allows a federated learning network to achieve consensus in a collective learning problem even in the presence of Byzantine devices. While blockchain technology was originally developed as a peer-to-peer financial system in the context of the cryptocurrency Bitcoin [3], recently there have been proposals for using blockchain technology as a distributed computing platform where arbitrary programs (blockchain-based smart contracts) can be run. The best known example of such a platform is Ethereum [4,5]. Blockchain-based smart contracts allow decentralized systems with mutually distrusting nodes to agree on the outcome of the programs. We provide the first proof-of-concept for using blockchain technology in federated learning applications. We do so by laying the foundation of a secure general framework for addressing collective learning problems.

Our approach is based on the federated learning scenario of McMahan et al. [2]. Via blockchain technology, we add a security layer on top of the classical approach that allows for taking care of the presence of Byzantine devices. Our blockchain approach also allows for logging events in a tamper-proof way: these logs can then be used, if necessary, to analyze the behavior of the devices in the network without incurring the risk that some malicious device has modified them. In addition, it provides a new way to understand how we debug and how we can approach data forensics in decentralized systems such as federated learning network. We use the simulator to vary the number of Byzantine devices and compare the performance – in terms of consensus time and probability of a correct outcome – of McMahan et al.'s [2] strategies and our blockchain-based variants both in the presence and in the absence of Byzantine devices.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 presents our proposed blockchain-based privacy-preserving deep learning framework. Section 4 evaluates the performance of the approaches through experiments in simulation. Section 4.5 discusses advantages and disadvantages of our blockchain approaches. Section 5 presents our conclusions and provides directions for future work.

2 Related Work

2.1 Privacy in Deep Learning

Deep learning aims to extract complex features from high-dimensional data and use them to build a model that relates inputs to outputs (e.g., classes). Deep learning architectures are usually constructed as multi-layer networks so that more abstract features are computed as nonlinear functions of lower-level features. Deep learning has been shown to outperform traditional techniques for speech recognition [8,9], image recognition [10,11] and face detection [12].

The existing literature on privacy protection in machine learning mostly targets conventional machine learning algorithms, as opposed to deep learning, and addresses **three objectives**: *(i) privacy of the data used for learning a model or as input to an existing model, (ii) privacy of the model, and (iii) privacy of the model's output.*

(1) Techniques based on **Secure Multi-party Computation (SMC)** can help protect intermediate steps of the computation, such as decision trees [13], linear regression functions [14], association rules [15], Naive Bayes classifiers [16], and k-means clustering [17], when multiple parties perform collaborative machine learning on their proprietary inputs. But SMC techniques impose non-trivial performance overheads and their application to privacy-preserving deep learning remains an open problem.

(2) Techniques based on **differential privacy** have been proposed to guarantee the confidentiality of personal data while training a differentially-private model [18–25]. 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.

(3) Techniques based on **privacy-preserving distributed learning** have been proposed to learn information from data owned by different entities without disclosing either the data or the entities in the data. Shokri and Shmatikov [26] and McMahan et al. [2] 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. [27] and Papernot et al. [28], where privacy-preserving models are learned locally from disjoint datasets, and then combined on a privacy-preserving fashion. However, the privacy guarantees of some of these solutions have recently been called into question [29].

Unlike previously proposed techniques, our system achieves all three privacy objectives in the context of collaborative neural-network training: it protects privacy of the training data, enables participants to control the learning objective and how much to reveal about their individual models, and lets them apply the jointly learned model to their own inputs without revealing the inputs or the outputs. 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 [30]. Recent work have demonstrated the feasibility of running complex deep learning inferences on local devices such as smartphones [31]. While in these works models are previously trained in an offline manner, our experiments proved that both the inference and the local retraining can be performed locally on a low-power device in a timely manner.

2.2 Blockchain

Blockchain is a peer-to-peer distributed ledger technology that was initially used in the financial industry [3]. The blockchain, a chronological ledger of transactions that ensures the integrity of the information included, can be used to capture and log both queries and its correspondent answers. Blockchain 2.0 introduces the concept of smart contracts [32], which is no longer limited to transactions between currencies, and there will be more extensive instruction embedded in the blockchain. The smart contract does not need mutual trust, as it is not only defined by the code, but executed by the code. Besides, it's completely automatic and cannot be intervened.

(1) Blockchain storage model, with non-tampering feature and traceability, ensures the privacy and credibility of the data.
(2) The smart contract that automatically execute the default instruction and the complete de-centric model guarantee the security of data sharing.
(3) Establish a reliable big data distribution system without trusting third parties.

The properties of blockchain make it a promising tool in many privacy informatics applications [33]: from building decentralized backbones for data exchange and interoperability, protocols enforced by immutable ledgers that keep track of data usage [35], and data provenance [36,37], to maintain user's privacy and security through the persistence of consent statements in blockchain [38]. Moreover, the blockchain technology offers practical means to safely and securely store and track the use of personal data as well as the parameters of the deep learning models. This increases the users' trust in the system and provides a rich source of information that can be used to better design future services. Our proposed framework deploys blockchain technology where anyone can read and validate transaction entries, but only authorized entities are able to create or write transaction to the blockchain.

3 Proposed Architecture

As shown in Fig. 1, our system design three major parts, namely, mobile devices (i.e. participants), hub, and blockchain network. The data producers collect the massive data through the smart contract to store in the blockchain, for the use of data sharing. The smart contract code runs on the contract layer of blockchain, which provides the authority to control the system.

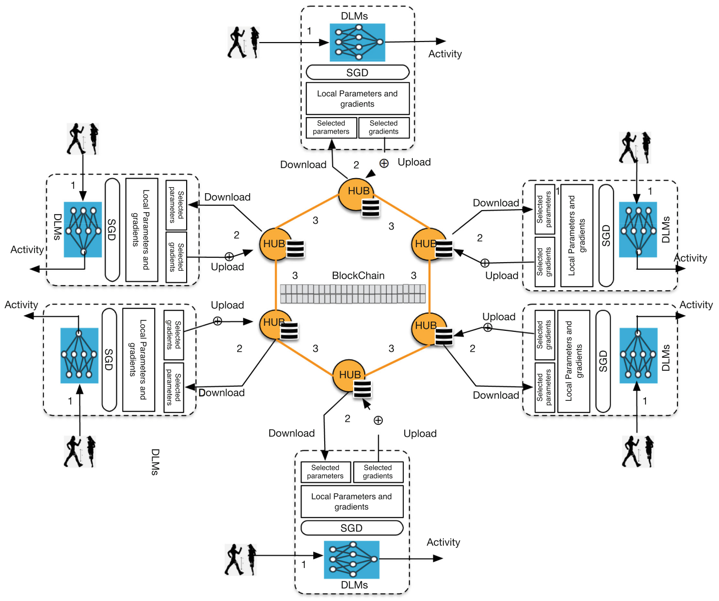

Fig. 1. High-level architecture of our deep learning system for activity recognition.

3.1 Participants

① **Inference.** We assume that each participant (R_i) relies on the deep learning models (DLMs), M_i, when conducting a inference. Specifically, R_i checkouts M_i from its local hub and uses it to conduct the inference. During this, R_i stores the information captured by its sensors, producing the inference data ($\mathbf{ID}_i$). $\mathbf{ID}_i$ allow R_i to update/improve the existing M_i. Specifically, a supervised machine learning approach is adopted: $\mathbf{ID}_i$ are used as input to M_i, while the class scores $\mathbf{O}_i$ (e.g. human activities) as the target output. Then, the fine-tunning of the DLM parameters $\mathbf{W}_i$ to the newly acquired data $\mathbf{ID}_i$ is accomplished using the standard back-propagation technique and by selecting an optimizer. With these new parameters, the participant is expected to increase its competences and adaptability to target inference.

Algorithm 1. Inference (based on CNN forward and backward propagation.)

```
Require: M-dimensional data, $\mathbf{ID_i} = [I_1, \cdots, I_n]^T$
Ensure: The class scores, $\mathbf{O_i} = [O_1, \cdots, O_c]$
 1: Download $\theta \times |\mathbf{w}^{(i)}|$ parameters from local hub.
 2: for $l := 1 \rightarrow \#HiddenLayers$ do
 3:     for $i := 1 \rightarrow \#RowunitinLayerl$ do
 4:         for $j := 1 \rightarrow \#ColumnunitinLayerl$ do
 5:             Find the layer activations by,
 6:             $O^l_{ij} = \varphi(a^l_{ij} w^l_{ij}) + b^l_{ij}$
 7:             Compute next layer inputs
 8:         end for
 9:     end for
10: end for
11: Keep the final output as $O^l$
12: Calculate error at the output layer.
13: for $l := 1 \rightarrow \#HiddenLayers$ do
14:     Find error partial derivation.
15:     Find error at the previous layer.
16: end for
17: Calculate the gradient of the error.
```

3.2 Hubs

② **Re-training.** After the inference, the data (ID) and class score ($\mathbf{O}$) are used together to improve the inference by re-training their DLMs (Ms). This can be done on the data-processing servers of the local hub. To this end, different learning strategies can be adopted. For instance, the participants (Rs) can create, store, and update Ms after each inference, or after a sufficient amount of ID has been collected. The design of specific learning strategies depends on the inference type. Then, these new Ms are committed and stored in the local repository, similar to the version control systems, of the local hub. Once created and stored, it is assumed that these new Ms cannot be used to recreate the raw ID of R. Therefore, all personal data remain safe as they do not leave the hub. Furthermore, they are deleted after all Ms are updated.

The stored Ms are further deployed locally and assigned a cumulative score S based on the $\mathbf{O}$ derived by validation of this model within new local inferences. As a result, a new candidate model (M_C) along with its performance score (S) is then created and locked on the local hub. To allow knowledge sharing – one of the key ingredients of the framework – this new M_C is then evaluated by the participant peers operating at other sites, i.e., different hubs within a network. To this end, the local hub publishes the changes (e.g., the difference between the previous version of M_j and M_C). Finally, the hub announces the update to the entire network. The goal of this is to assure a fair validation of the M_C before it can be adopted as the new version of the Ms for target inference.

Algorithm 2. Re-training (based on Distributed selective SGD)

```
Require: Choose initial parameters $\mathbf{w}^{(i)}$ and learning rate $\alpha$.
1: repeat
2:     Get $\theta \times |\mathbf{w}^{(i)}|$ parameters from the blockchain and replace the corresponding local parameters.
3:     Run stochastic gradient descent (SGD) on the local dataset and update the local parameters $\mathbf{w}^{(i)}$.
4:     Compute gradient vector $\triangle\mathbf{w}^{(i)}$ which is the vector of changes in all local parameters due to SGD.
5:     Store $\triangle\mathbf{w}^{(i)}$ to the blockchain, where $S$ is the set of indices of at most $\theta_u \times |\mathbf{w}^{(i)}|$ gradients
6: until an approximate minimum is obtained
```

3.3 Parameter Blockchain

③ Updates. I. Publish. First, the source participant ($R_{(s)}$) 'advertises' the new candidate model (M_C^s) by announcing the DLM updates to the entire network. Then, the destination participants ($R_{(d,i)}$), where $i = 1 \cdots N$ denotes the target hubs, are notified by their local hub that there is an update available in the network. This can be achieved by the subscription pipeline they have with their local hub. In case the updates are available, the participants can retrieve and apply them to their working directory. Once $M_{(C)}^s$ is adopted from the source hub, $R_{(d,i)}$ starts its evaluation, quantified in terms of the scores ($S_{(R_{(d,i)},\, M_{(C)}^s)}$). Additionally, in order to leverage the new local data, the destination hubs can also return the model updates to the source hub (obtained in a similar fashion as when creating the M_C). These, in turn can be used to construct the new model at the source hub (Fig. 2).

II. Validate. During the next stage, $R_{(s)}$ is to consolidate the feedback information from the destination hubs. This can be achieved using time-constraints (i.e., waiting for a pre-defined period of time to receive the feedback), and/or when a target consensus is achieved. If the score for the M_C is higher than for the currently accepted model (M_s), i.e., $\overline{S_{(i,M_C^s)}} > \overline{S_{(i,M_s)}}$, R_s creates a new model $M_{(s+1)}$, which is then published to all connected hubs, and committed to their participants' local repositories. In this way, the new baseline model for future inferences is endorsed by the network. This process can be implemented via modern control version systems in order to store and share the resulting model configurations (e.g., the DLM topology, hyper-parameters, etc.) obtained after new inferences. This is an important feature of our framework since it allows the participants to rollback to the last consensual version of M_{s+1}, in case a consensus did not take place within the network. Moreover, since the participants keep the list of all changes in their local repository, there is a promising research approach in analyzing the metadata available in the updates applied to the repository.

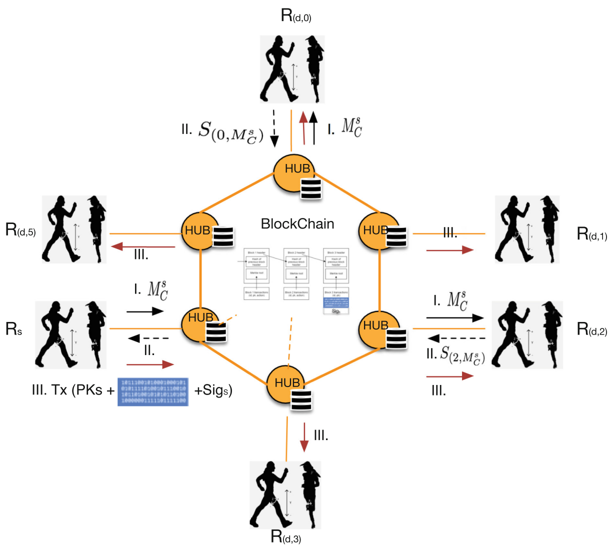

Fig. 2. A BlockChain-based deep learning network.

III. Consensus. In order to notarize and log the creation of the new models and their consensus processes within the network, $R_{(s)}$ is required to send a transaction to a blockchain including information such as the timestamp of the global update broadcast, a hash string that encapsulates the information about the model update (e.g., differences in the weights, hyper-parameters, etc.), and an encrypted data field signed by $R_{(s)}$ containing information such as the public IDs of the participants that took part in the consensus process and their correspondent feedback scores. This transaction on the blockchain is necessary to allow participants within the network to prove/validate how models were created, who participated in their consensus process, and when did those transaction take place. It is also important to highlight that the hash string included in this transaction is useful to check and confirm that the models acquired by participants are indeed the version agreed upon by the network, and not a corrupted version from a third-party agent. In addition, the encrypted data field signed by $R_{(s)}$ contain sufficient information to allow the users to confirm their participation in the consensus process and check that $R_{(s)}$ was not biased at the moment of promoting $M_{(j)}$. This information is readable by any participant in the network, since they know the public identifier (i.e., public key) of $R_{(s)}$. By contrast, the peers connected to the blockchain but not members of the private clinical network do not have the means to decrypt this information or identify the nodes involved in the consensus process. This is because all required public/private keys remain within the boundaries of the clinical private network. Finally, note that the whole consensus reaching process could have been implemented directly on the blockchain via 'smart contracts' in order to prevent intruders from 'attacking' the network; yet, we assume here that the network access is protected.

4 Evaluation

4.1 Datasets and Learning Objectives

As Fig. 3, 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, 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 s of accelerometer samples as the $d = 43$ dimensional features in our models. Our final sample contains 5,418 accelerometer traces from 35 users, with on average 150.504 traces per user and standard deviation of 44.73.

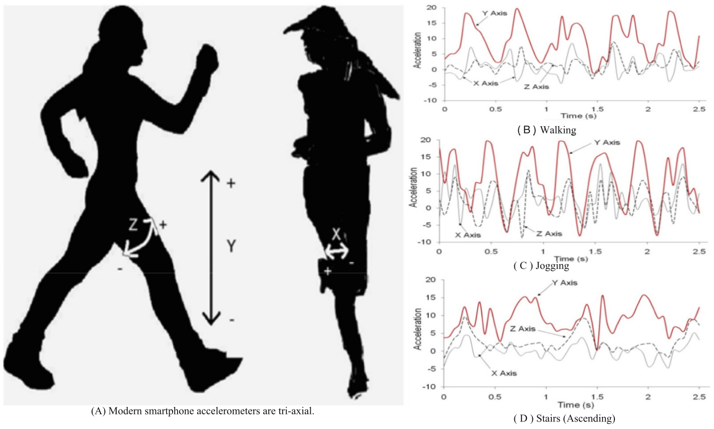

Fig. 3. (A) Modern smartphone accelerometers are tri-axial. (B–D) Accelerometer graphs for three dynamic activities.

4.2 Computing Framework

We use Torch7 and Torch7 *nn* packages. This popular deep-learning library has been used and extended by major Internet companies such as Facebook.

4.3 Neural Network Architectures

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 4 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.

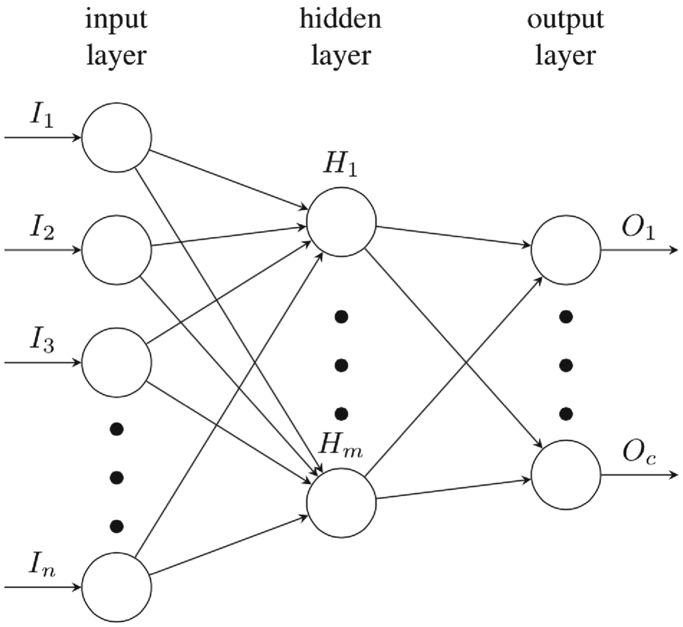

Fig. 4. The architecture of the two-layer feed-forward network.

4.4 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 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. The training and testing were performed with 5-fold cross validation, using early stopping as well as l2-regularisation to prevent overfitting. Each neuron's weight in the *shared* and *local* models was initialised randomly from $N(0,1)/\sqrt{2.0/n}$, 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 optimal 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 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.

4.5 Results

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 *local* 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. We also observe that, after on average 163 samples, the *local* model performs better than the personal model. This means that the user would need to perform and label, on average, 163 activities in order to get a local model that outperforms her personal one. 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.

5 Conclusions and Future Work

Creating and training of deep learning models have been a computationally-hungry process. So the adoption of these models in embedded devices has been a challenging task, especially for low-cost mobile devices. Federated learning, one decentralized approach, learns a shared model by aggregating locally-computed updates, and leaves the training data distributed on the mobile devices. The federated learning network interconnecting the participants' devices is not public. Thus, it requires a permission of one or several parties to join and start contributing to the deep learning process. We understand that some form of trust is required in the institutions that deal with a sensitive task.

In this paper we proposed the first deep learning approach for tackling the privacy issues in the use of personal data by mobile devices. We illustrated this framework using activity recognition as an example which is conducted simultaneously in multiple mobile devices. While the approach proposed here offers the main principles for secure data and models sharing between multiple mobile devices. To this end, we aim to collect the data from several parties in order to test the framework in a data-exchange scenario. Then, we aim to run in real time on private network to enable efficient and real-time learning of models. For this, existing deep learning approaches will need to be adapted so that they can efficiently be integrated with the blockchain technologies. Also, These are topics of our ongoing research.

References

1. Poushter, J.: Smartphone ownership and internet usage continues to climb in emerging economies. Pew Research Center Report (2016)
2. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., et al.: Communication-efficient learning of deep networks from decentralized data (2016). arXiv preprint: arXiv:1602.05629
3. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf
4. Buterin, V.: A next-generation smart contract and decentralized application platform. Ethereum project white paper (2014). https://github.com/ethereum/wiki/wiki/White-Paper
5. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum project yellow paper (2014). http://gavwood.com/paper.pdf
6. Hannun, A., Case, C., Casper, J., et al.: DeepSpeech: scaling up end-to-end speech recognition (2014). arXiv:1412.5567
7. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on ImageNet classification (2015). arXiv:1502.01852
8. Graves, A., Mohamed, A.R., Hinton, G.: Speech recognition with deep recurrent neural networks. In: ICASSP (2013)
9. Hinton, G., Deng, L., Yu, D., Dahl, G., et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012)
10. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012)

11. Simard, P., Steinkraus, D., Platt, J.: Best practices for convolutional neural networks applied to visual document analysis. In: Document Analysis and Recognition (2013)
12. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: DeepFace: closing the gap to human-level performance in face verification. In: CVPR (2014)
13. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_3
14. Du, W., Han, Y., Chen, S.: Privacy-preserving multivariate statistical analysis: linear regression and classification. In: SDM, vol. 4, pp. 222–233 (2004)
15. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: KDD (2002)
16. Vaidya, J., Kantarcoğlu, M., Clifton, C.: Privacy-preserving Naive Bayes classification. VLDB **17**(4), 879–898 (2008)
17. Jagannathan, G., Wright, R.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: KDD (2005)
18. Dwork, C., Rothblum, G., Vadhan, S.: Boosting and differential privacy. In: FOCS (2010)
19. Chaudhuri, K., Sarwate, A., Sinha, K.: A near-optimal algorithm for differentially-private principal components. JMLR **14**(1), 2905–2943 (2013)
20. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In: NIPS (2009)
21. Zhang, J., Zhang, Z., Xiao, X., Yang, Y., Winslett, M.: Functional mechanism: regression analysis under differential privacy. VLDB **5**(11), 1364–1375 (2012)
22. Rubinstein, B., Bartlett, P., Huang, L., Taft, N.: Learning in a large function space: privacy-preserving mechanisms for SVM learning. J. Priv. Confidentiality **4**(1), 4 (2012)
23. Sarwate, A., Chaudhuri, K.: Signal processing and machine learning with differential privacy: algorithms and challenges for continuous data. IEEE Signal Process. Mag. **30**(5), 86–94 (2013)
24. Chaudhuri, K., Monteleoni, C., Sarwate, A.: Differentially private empirical risk minimization. JMLR **12**, 1069–1109 (2011)
25. Wainwright, M., Jordan, M., Duchi, J.: Privacy aware learning. In: NIPS (2012)
26. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1310–1321. ACM (2015)
27. Hamm, J., Cao, P., Belkin, M.: Learning privately from multiparty data. In: Proceedings of the 33rd International Conference on Machine Learning, pp. 555–563 (2016)
28. Papernot, N., Abadi, M., Erlingsson, U., Goodfellow, I., Talwar, K.: Semi-supervised knowledge transfer for deep learning from private training data. In: Proceedings of the 5th International Conference on Learning Representations (2017)
29. Hitaj, B., Ateniese, G., Pérez-Cruz, F.: Deep models under the GAN: information leakage from collaborative deep learning. CoRR, vol. abs/1702.07464 (2017)
30. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
31. Georgiev, P., Lane, N.D., Rachuri, K.K., Mascolo, C.: DSP.Ear: leveraging co-processor support for continuous audio sensing on smartphones. In: Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, pp. 295–309. ACM (2014)

32. Peters, G.W., Panayi, E.: Understanding modern banking ledgers through blockchain technologies: future of transaction processing and smart contracts on the internet of money. In: Tasca, P., Aste, T., Pelizzon, L., Perony, N. (eds.) Banking Beyond Banks and Money. NEW, pp. 239–278. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42448-4_13
33. Kuo, T.T., Kim, H.-E., Ohno-Machado, L.: Blockchain distributed ledger technologies for biomedical and health care applications. J. Am. Med. Inform. Assoc. **24**(6), 1211–1220 (2017)
34. Kuo, T.T., Ohno-Machado, L.: ModelChain: decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks (2018). arXiv preprint: arXiv:1802.01746
35. Topol, E.J.: Money back guarantees for non-reproducible results? BMJ **353**, i2770 (2016)
36. Baxendale, G.: Can blockchain revolutionise EPRs? ITNOW **58**(1), 38–39 (2016)
37. Taylor, P.: Applying blockchain technology to medicine traceability (2016)
38. Brodersen, C., Kalis, B., Leong, C., et al.: Applying blockchain technology to medicine traceability (2016)