Gutierrez, N. [et al.]. Privacy preserving deep learning framework in fog computing. A: International Conference on Machine Learning, Optimization, and Data Science. "Machine Learning, Optimization, and Data Science, 6th International Conference, LOD 2020: Siena, Italy, July 19-23, 2020: revised selected papers, part I". Berlín: Springer, 2020, p. 504-515. ISBN 978-3-030-64583-0.
The final authenticated version is available online at https://doi.org/10.1007/978-3-030-64583-0_45

Privacy preserving Deep Learning framework in Fog computing

Norma Gutiérrez, Eva Rodríguez, Sergi Mus, Beatriz Otero, and Ramón Canal

Computer Architecture Department, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{norma,evar,smus,botero,rcanal}@ac.upc.edu

Abstract. Nowadays, the widespread use of mobile devices has raised serious cybersecurity challenges. Mobile services and applications use deep learning (DL) models for the modelling, classification and recognition of complex data, such as images, audio, video or text. Users benefit from the wide range of services and applications offered by these devices but pay an enormous price, the privacy of their personal data. Mobile services collect all different types of users' data, including sensitive personal data, photos, videos, clinical data, banking data, etc. All this data is pooled to the Cloud to train global DL models, and big companies benefit from all the collected users' data, posing obvious serious privacy issues.
This paper proposes a privacy preserving framework for Fog computing environments, which adopts a distributed deep learning approach. Internet of Things (IoT) end nodes never reveals their sensitivity to the Cloud server; instead, they share a fraction of the users' data, blurred with Gaussian noise, with a nearby Fog node. The DL methods considered in this work are Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN), and for both cases the accuracy is similar to the centralized and privacy violating approach, obtaining the best results for the CNN model.

Keywords: Cyber-attacks, Privacy, Deep Learning, Machine Learning, Security, Fog computing, IoT

1 Introduction

Nowadays, mobile devices have become essential in people's daily lives. Users benefit from a wide range of services and applications that makes use of DL methods with high accuracy on the modelling, classification and recognition of complex data, as images, audio, video or text. These DL-based services and applications include recommendation systems, speech and image recognition, target advertising, health monitoring, etc. Most of these services are for free, but they collect high sensitive users' data, such as personal users' data, photos, videos, and even users' banking data. The widespread usage of these services and applications has raised important cybersecurity challenges, especially privacy issues for users' personal data. Individuals are con-

cerned about their collected data being used for unintended purposes, for example the data collected by e-healthcare systems for real-time diagnosis and medical consultancy, or even the data collected by face recognition applications that can be later used for targeted social advertising.The principal beneficiaries are companies that collect massive amounts of data from their users, since the success of existing DL techniques highly depends on the amount of available data for training the network. The centralized Cloud approach does not respect the privacy and confidentiality of users' data and also, it is staring to result inefficient due to limited computation resources and bandwidth. This has led to the adoption of distributed DL [1] to overcome response delays, computation bottlenecks, as well as privacy issues preventing users from pooling their data to a central server. On the one hand, the exchange of information with the server poses evident privacy issues. On the other hand, DL trained models incorporate essential information of its training data sets, and then it is not complicated to extract sensitive information from DL classifiers.

Fog computing brings Cloud Computing closer to the physical world of smart things. It was Cisco who initially conceived Fog computing as a Cloud computing extension to the edge of an enterprise network. Continuous innovations in hardware and software have made possible transferring computation to the edge of the networks. Currently, Cloud based architectures are used to process and store IoT devices' generated data, but fog computing paradigm is a promising solution to scale and optimize IoT infrastructures. Fog architectures consist of three layers, unlike cloud-based solutions that consider two layers, introducing a fog computing layer that can be divided into multiple abstraction sub-layers, between the Cloud and the IoT end devices. Edge nodes will provide computing, storage, communication, control and security services overcoming centralized Cloud based approaches issues. Fog computing considers distributed DL, where IoT devices, as well as Fog devices perform part of the learning process. IoT devices can conduct part of the learning process thanks to lightweight DL libraries developed by industrial companies, as TensorFlow Lite [2]. However, it is not realistic to suppose that all IoT devices are equipped with substantial computational resources, since most of them have low-spec chips. Moreover, end users would not wish to share their local models for privacy concerns. All this led us to devise a framework that considers DL models running on Fog devices, not in end devices. IoT end devices will send a fraction of their data, blurred with Gaussian noise, to nearby fog nodes, where DL algorithms will be executed.

Lot of work has been done in the privacy area, mainly based on differential privacy, introduced by Dwork [3] in 2006. Differential privacy guarantees that the models or algorithms learnt from participants released information, but without participants' data being included. Then, it will not be possible to determine the specific participant information used in computation. Differential privacy is used in DL frameworks to avoid disclosure of private information. The algorithms proposed in [4, 5] were based on a differentially private version of stochastic gradient descent (SGD). Hitaj et al. [6] continued this work, training DL structures locally and only sharing a subset of the parameters, obfuscated via differential privacy. But all these works do not consider a distributed DL architecture, they consider a central server to build a global model which combines all the users' data or provides all the parameters to end users.

In this paper, we focus on privacy and confidentiality concerns proposing a DL privacy preservation framework for IoT. This framework preserves users' privacy performing distributed DL. In this way, IoT nodes never reveal their sensitive data to the Cloud server, they only share a fraction their sensitive data, blurred with Gaussian noise, with a nearby Fog node, which subsequently computes the gradient of the users' data and updates a fraction of them to the Cloud. To evaluate the results obtained the framework is compared with centralized privacy-violating solutions, considering both MLP and CNN models, since they are the most widely used for DL privacy preservation.

2 Related Work

Centralized deep learning poses serious privacy threats, since users pool their data in a central server that train a global model, which combines data from all participants. First works that address privacy issues, proposed a privacy preserving model. Shokri and Shmatikov [1] propose a distributed training technique to preserve users' privacy based on selective stochastic gradient descent (SGD). The authors take advantage of the fact that optimization algorithms used in DL can be parallelized and executed asynchronously. Then, users can train their own datasets independently and only share a small subset of the key parameters of their models. This methodology works for all different types of DL models guaranteeing users privacy, while maintaining accuracy levels. The solution was evaluated using MLP and CNN models and two datasets MNIST [7] and SVHN [8], typically used for image classification. For the MNIST dataset, the system achieved an accuracy of 99.14% when participants share the 10% of their data, close to the accuracy (99.17%) obtained for a centralized privacy violating model approach. While for the SVHN dataset, the accuracy was of 93.12%. Pong et al. [9] continued this work improving the system in two ways: first, data is not leaked to the server, and second, homomorphic encryption is used providing more security to the system without compromising its accuracy.

A similar approach has been undertaken in Fog computing environments. Lyu et al. [10] defined a Fog embedded privacy preserving deep learning framework (FPPDL) to reduce computation and communication costs while preserving privacy. Privacy is preserved by a two-level protection mechanism, which first uses Random Projection (RP) to protect privacy, perturbing original data, but preserving certain statistical characteristics of the original data, and then Fog nodes train differentially fog-level models applying Differentially Private SGD. The framework was implemented using MLP with two hidden layers, using ReLU activations, and it was tested using three different datasets typically used in image classification: MNIST [7], SVHN [8] and multiview multicamera dataset [11]. The solution achieved an accuracy of 93.31% for the MNIST dataset and 84.27% for the SVHN dataset, almost 10% lower than for the centralized framework, but reducing communication and computation costs significantly.

The work presented in this paper proposes a framework that preserves users' privacy performing distributed DL in Fog computing, exploring two models: MLP [12] and

CNN [13], since they are the models most commonly used for privacy preservation [1, 4, 9, 10, 14]. The framework is evaluated using a digits' dataset. Decentralized protection techniques are applied to this dataset on the Fog to classify the digit, preserving its original value correctly. The information used in this dataset can be related to the telephone number of an individual of the digits of his DNI.

The framework proposed in this paper differs from [1, 9] because we only share the gradients with the Fog nodes, and we inject Gaussian noise to improve users' privacy. And from [10] because we also consider the CNN model, not just MLP, obtaining better results for CNN. The framework proposed in this paper achieves better results for CNN. Another key aspect, that differentiates our framework from all previous works [1, 9, 10], is that we perform a validation process in the server when updating the mean gradients, which lead our framework to obtain better accuracy than in previous works.

3 Framework

This section describes the privacy preserving DL framework proposed in this paper. A three-level framework is proposed as depicted in Fig. 1, which consists of numerous IoT devices at the bottom level (*V* devices), or end nodes, Fog nodes in the middle level (*N* fog nodes) and the Cloud server at the top level.

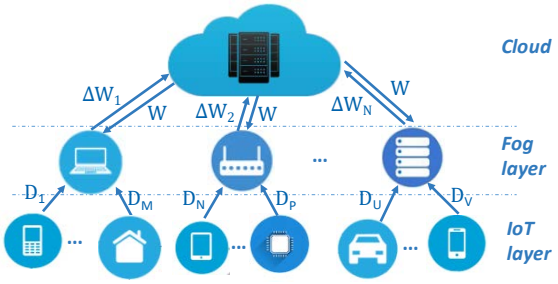

Fig. 1. Fog-embedded architecture.

The framework in this architecture has been designed to preserve users' privacy. Therefore the IoT end nodes never reveal their data to the Cloud server by only sharing part of their data with the nearby fog node. Moreover, Gaussian noise has been injected to provide higher privacy to users' sensitive data. The devices at the end nodes contain all the data to train the model. Each device selects a specified portion of the dataset and preprocesses the data. This preprocessing consists of loading the raw data from the dataset, separate it into labels and 3D images, which will later be flattened in the model. Then, a selected amount of data will be stored in a pickle which will be sent to the fog node. Once the data is selected and preprocessed, it is sent to its' associated fog node. Each fog node has a selected number of end nodes associated with it; in our experiments, a total of 6 end nodes were selected for each fog node. Once all the preprocessed data from the different IoT devices is in the fog node, it

downloads a copy of the latest version of the model from the Cloud server. Initially, the model weights have a random initialization ($\overrightarrow{W}$, or random weight vector). When the fog node receives the data and the model it starts the training process with the specified epochs. In order to share as little as possible to the Cloud, once each epoch is completed only the gradient between the initial weights and the new weights, $\overrightarrow{\Delta W}$, is passed to the Cloud, thus preserving the client's privacy by only passing a little portion of the total results.

The Cloud server validates the received updates with a portion of the reserved dataset called validation data which is stored in the Cloud server, and stores them in $\overrightarrow{W}$ as $\overrightarrow{W} = \overrightarrow{W} + \overrightarrow{\Delta W}$. After the framework is collaboratively trained, each Fog node can test and evaluate its data independently.

The privacy preserving framework is compared with a centralized privacy violating Cloud architecture (see Fig. 2), where all the users' data is uploaded to the server, and the global model is trained using the same metrics as before. That is to say, devices, or end nodes, only upload raw data to the Cloud, who is responsible for the preprocessing, training, and validation of the model. In the fog-embedded architecture, not only the data is preprocessed in the end nodes, but the fog nodes train the model leaving the Cloud server with one task, to validate and update the model to be accessible for other fog nodes. We have chosen MLP and CNN for the fog-embedded architecture since these are the best architectures and most commonly used to analyze the selected problem. It is noteworthy that the accuracy results obtained for the privacy preserving framework proposed in this paper are similar to the achieved by the centralized framework, as detailed in the next section. That is, privacy is preserved achieving similar accuracy results. In Sect. 5, it will be shown that the models have less accuracy in the fog-embedded architecture. However, privacy is much higher than the centralized architecture, which violates end-users' privacy.

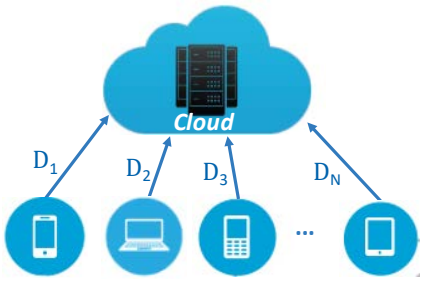

Fig. 2. Centralized architecture.

4 Performance Evaluation

4.1 Dataset

We evaluate our system on The Street View House Number (SVHN) dataset [8] that contains 73.257 digit images of 32 × 32 pixels for training, and 26.032 for testing.

The objective is to classify the input as one of ten possible digits in the integers {0,1,...,9}, so the size of the output layer is ten. From these samples, we used a total of 70.000 for training and 10.000 for validation. The validation samples reside in the Cloud server, whereas the training samples come from the different end nodes that each fog node has associated. In the fog-embedded architecture, we consider balanced data partitions for three, five, and seven fog nodes and its end nodes. Each fog node connects with six end nodes, and then each fog-level model is trained on 23.300, 14.000, and 10.000 samples depending if we are running three, five or seven fog nodes. This way, each end node owns 5%, 3.3%, and 2.4% data of the entire training. Decreasing the percentage of data that each end node provides to the fog node preserves the client's privacy since less data is demanded as the number of fog nodes is increased. Table 1 summarizes training and test examples used in this work for centralized and fog-embedded architectures.

Table 1. . Size of training and test SVHN dataset.

Architecture	Training	Testing
Centralized	70.000	10.000
Fog-embedded (with 3 fog nodes)	23.300 per fog node Each fog node has 6 clients, and each end node has 3.900 training samples	10.000
Fog-embedded (with 5 fog nodes)	14.000 per fog node Each fog node has 6 clients, and each end node has 2330 training samples	10.000
Fog-embedded (with 7 fog nodes)	10.000 per fog node Each fog node has 6 clients, and each end node has 1.700 training samples	10.000

4.2 Description models

This work proposes a framework that preserves users' privacy performing distributed DL in Fog computing using two models: Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN).

The MLP architecture consists of an input layer, one or more hidden layers and an output layer. In our model, we have a simple multilayer perceptron with two hidden layers with 256 and 128 units using `ReLU` activations, and a final `SoftMax` layer with 10 units. The exact description of the model used is below.

```
model.Sequential([
  model.Flatten(),
  model.Dense(1024, ReLU),
  model.Dense(1024 –> 256, ReLU),
  model.Dense(256 –> 128, ReLU),
  model.Dense(128–> 10, SoftMax)
])
```

The CNN architecture consists of three different types of layers: convolutional pooling and classification. In the implementation, the CNN model consists of a simple Convolutional Neural Network with also, two hidden layers with 128 and 64 units respectively, since our main goal was to maintain the structure of the model in both cases as similar as possible. In our hidden layers, each layer consists of a convolutional layer and a pooling layer. The model also included a `SoftMax` layer with 10 units. The detailed model is in the diagram below.

```
model.Sequential([
  model. Conv2D(256, ReLU),
  model.MaxPooling2D(pool_size),
  model.Conv2D(256 –> 128, ReLU),
  model.MaxPooling2D(pool_size),
  model.Conv2D(128–> 64, ReLU),
  model.MaxPooling2D(pool_size),
  model. Flatten(),
  model.Dense(64–> 10, SoftMax)
])
```

Finally, for both implementations, other relevant parameters were considered as SGD (Stochastic gradient descent) with a learning rate of 0.1 and a categorical cross entropy loss.

4.3 Experiments

The network has been trained using a 1.6 GHz Dual-Core Intel Core i5 with 8 GB of memory and an Intel UHD Graphics 617 1536 MB, using Tensorflow version 2.0.0 with Python 3.7.4 version and Numpy library. Experiments using MLP and CNN were executed in a Cloud environment, specifically Microsoft Azure. It was used the virtual machine STANDARD_D14_V2. We used a callback to register all the statistics of each execution, enabling us to compare each model easily and storing all the data in a logs folder.

Moreover, we normalized the dataset by subtracting the average and dividing by the standard deviation of data training samples. After that, we changed the data format converting the images into Numpy arrays and stored them a file. Finally, in the fog-embedded architecture we applied 1% of Gaussian noise to the samples to preserve their privacy.

As stated before each data end-node sends a different portion of the dataset to the fog node and the fog node trains the model a stated number of epochs. In MLP a total of 100 epochs were chosen, and in CNN, a total of 20 epochs were chosen. These numbers are a good compromise between an optimal convergence and the improvement of the accuracy since if the number of epochs was to be increased; the accuracy progressed poorly and consumed many resources.

Once the model is trained, only the mean value of each weight of every epoch is passed to the Cloud server allowing thus, to preserve the user's privacy, since no raw

data was sent, only a statistical value of the weight of each epoch. Then, when the Cloud server has received all the mean values from each fog-node, updates the model, as explained in Sect. 3. Finally, for each experiment in both architectures we have conducted 50 simulations and the results obtained were shown in the next section.

4.4 Results

During the training process, a Tensor board callback was used, which enabled to monitor the accuracy and loss over each epoch (see Fig. 3 and Fig. 4). In this way, the networks can be compared. After network training, we determine which architectures have major benefits, and those were kept while the underperforming ones were rejected. For both DL methods the mean-gradient was uploader for the server.
Experiments using the centralized architecture model and MLP reached an accuracy of 88.92% after 100 epochs (see Fig. 3), while the accuracy for CNN was 94.08% with only 20 epochs (see Fig. 4).

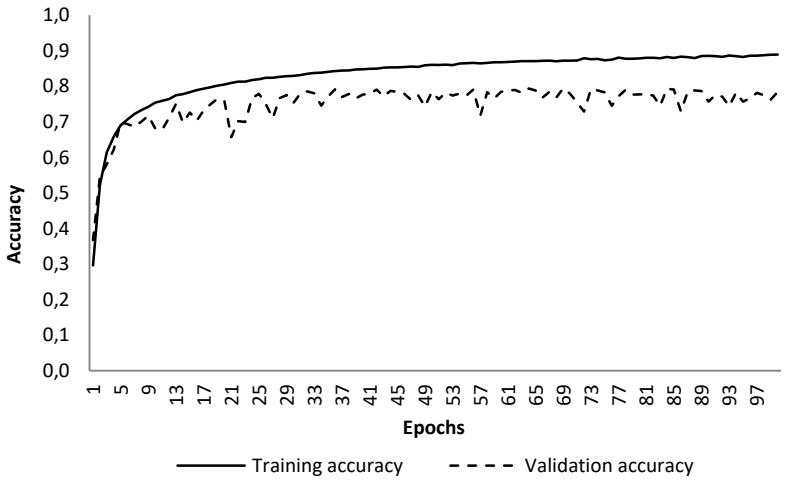

Fig. 3. Evolution of accuracy and validation accuracy during training using MLP in a centralized architecture (non-privacy data preserving).

Figures 5 and 6 show the results obtained for the experiments conducted on the fog-embedded architecture for both DL models using the SVHN dataset. As before, figures show the accuracy achieved in the training and validation processes. However, in this case the validation accuracy values are determined by getting the averaged value over all fog nodes. The graphs in both figures (MLP and CNN) show that accuracy for the fog-embedded architecture is slightly improved compared to the centralized model (see Figs. 3 and 4).
For all experiments in each scenario the percentage difference in loss between the training and validation dataset is smaller than 1%. This indicates that no overfitting takes place in any case.

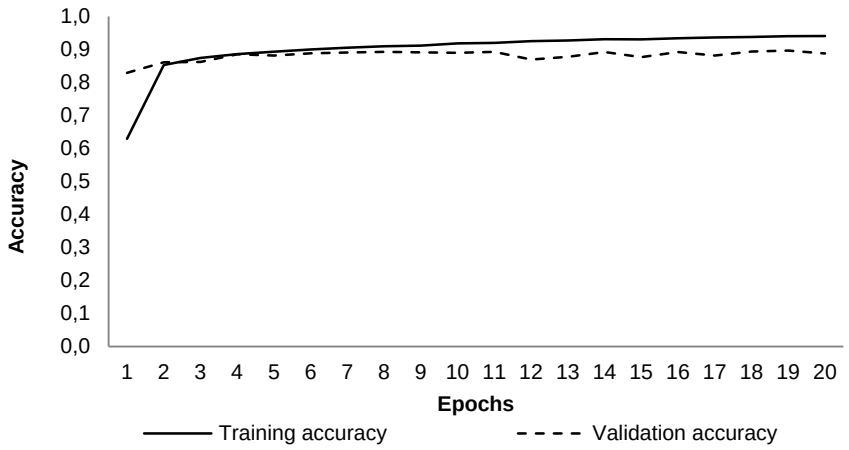

Fig. 4. Evolution of accuracy and validation accuracy during training using CNN in a centralized cloud architecture (non-privacy data preserving).

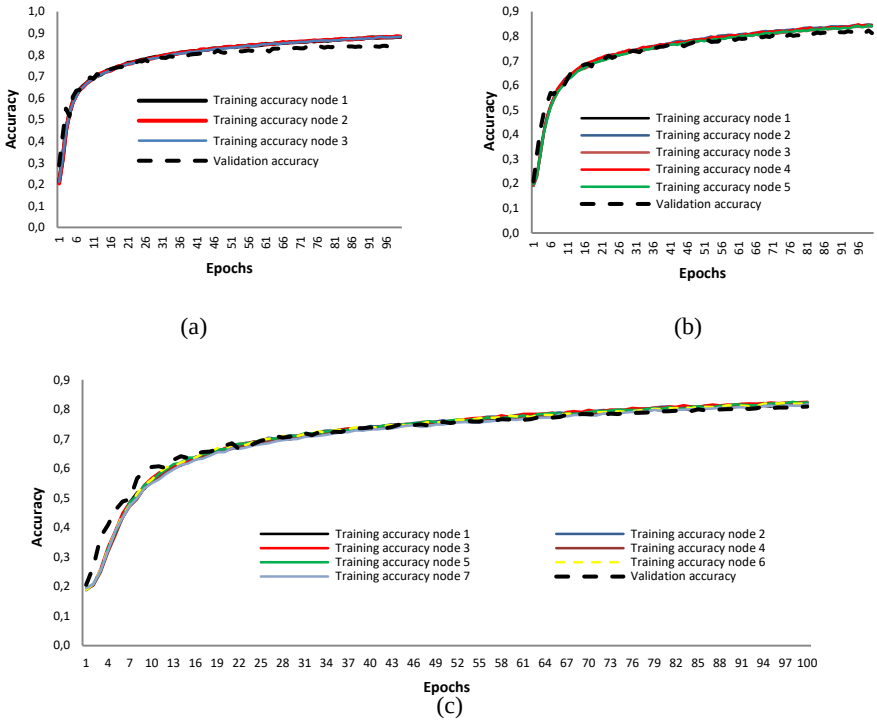

Fig. 5. Evolution of accuracy and validation accuracy during training using MLP in a fog-embedded architecture (privacy data preserving): (a) for 3 fog nodes; (b) for 5 fog nodes; (c) for 7 fog nodes.

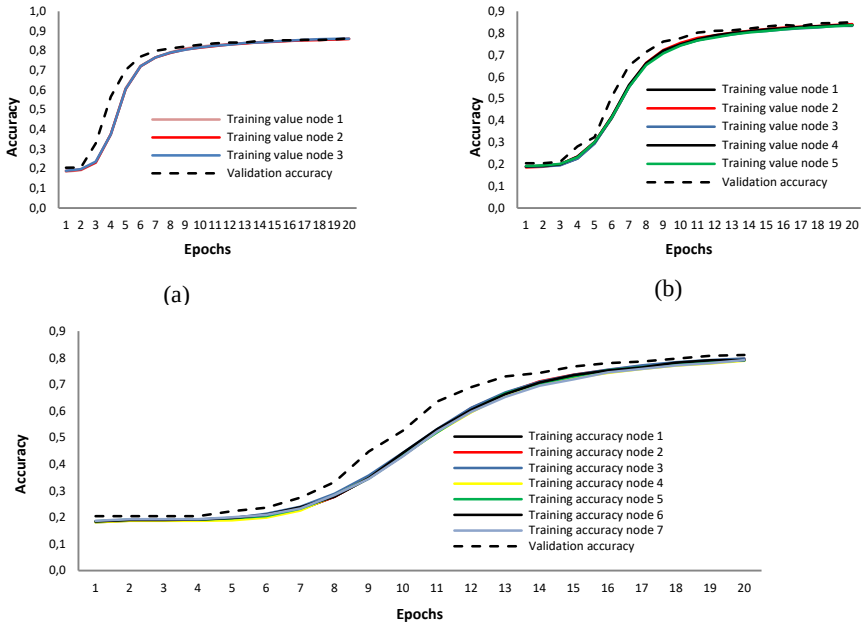

Fig. 6. Evolution of accuracy and validation accuracy during training using CNN in a fog-embedded architecture (privacy data preserving): (a) for 3 fog nodes; (b) for 5 fog nodes; (c) for 7 fog nodes.

Table 2. . Summary of the training and validation accuracy average with 100 epochs (to MLP architecture) and 20 epochs (to CNN architecture)

Architecture	**Framework**			
	MLP		**CNN**	
	Training	**Validation**	**Training**	**Validation**
Centralized	0,8892	0,7841	0,9408	0,8878
Fog-embedded (with 3 fog nodes)	0,8837	0,8357	0,8606	0,8624
Fog-embedded (with 5 fog nodes)	0,8426	0,8120	0,8370	0,8500
Fog-embedded (with 7 fog nodes)	0,8205	0,8092	0,7923	0,8104

Table 2 summarizes the obtained results in all experiments, where we can appreciate the difference between both architectures and models for validation and training. In this table, we observed the average values of the validation process to fog-embedded architecture is slightly close (or equal in the order of the tenths) to the training values to the same architecture in both models. This is due to in the fog-embedded architecture we have distributed 10.000 validation samples, and this amount of samples is

very similar to the training samples. Usually, we have a more significant amount of training samples than validation samples, such as in the centralized model. Also, Figs. 5 and 6 show the effect to increase the fog nodes number influences the decrease in precision. This is mainly because increasing the number of fog nodes decreases the number of samples for the training and testing processes, all this makes the precision of the model found in the training and validation phase less accurate.

5 Conclusion

This paper proposed a privacy preserving framework for Fog computing environments, which adopted a distributed deep learning approach. The proposed framework consists of three levels, with IoT devices at the bottom level, Fog nodes in the middle level and the Cloud server at the top level. Fog nodes trained efficiently the network combining fractions of end nodes data, preserving in this way users' privacy. Each fog node was connected to six clients, and the number of fog nodes varied between 3, 5 and 7. The experiments conducted considered 70.000 training samples, distributed uniformly between all the clients, and obtained similar accuracies to the centralized Cloud training privacy violating method. Accuracy between 80%-86% was achieved, and the results were similar to each other independent of the number of Fog nodes. The framework considered two different DL models: MLP and CNN. The experiments results carried out showed that the CNN model worked significantly better than MLP, being the most remarkable improvement the number of the epochs needed in each architecture, since CNN needed five times fewer epochs than MLP to achieve the same convergence.

Future work will include a formal statistical analysis the obtained results, and further word will be conducted for improving the framework exploring new ways to update the model with a portion of the gradients, and users' privacy protection mechanism will be improved applying Random Projection to the original data.

6 Acknowledgements

This work is partially supported by Generalitat de Catalunya under the SGR program (2017-SGR-962) and the RIS3CAT DRAC project (001-P-001723).

7 References

1. Shokri, R., and Shmatikov, V.: "Privacy-preserving deep learning", in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1310-1321, October 2015
2. TensorFlow Lite, https://www.tensorflow.org/lite
3. Dwork, C.: "Differential Privacy", in Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP), pp. 1-12, 2006

4. Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L.: "Deep learning with differential privacy", in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 308-318, October 2016
5. Dwork, C., and Roth, A.: "The algorithmic foundations of differential privacy", Journal Foundations and Trends in Theoretical Computer Science, vol. 9, no. 3-4, pp. 211-407, August 2014
6. Hitaj, B., Ateniese, G., and Perez-Cruz, F.: "Deep models under the GAN: Information leakage from collaborative deep learning", in Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 603-618, November 2017
7. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P.: "Gradient-based learning applied to document recognition", in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, November 1998
8. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A.: "Reading digits in natural images with unsupervised feature learning", in NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning, vol. 2011, no. 2, 2011, p. 5
9. Phong, L. T., Aono, Y., Hayashi, T., Wang, L., and Moriai, S.: "Privacy-preserving deep learning: Revisited and enhanced", in L. Batten, D. Kim, X. Zhang and G. Li (eds) Applications and Techniques in Information Security, Communications in Computer and Information Science, vol. 719, pp. 100-110, June 2017
10. Lyu, L. , Bezdek, J. C. , He, X., and Jin, J.: "Fog-Embedded Deep Learning for the Internet of Things", IEEE Transactions on Industrial Informatics, vol. 15, no. 7, pp. 4206-4215, April 2019
11. Roig, G., Boix, X., Shitrit, H. B., and Fua, P.: "Conditional random fields for multi-camera object detection", in Proceedings of the IEEE International Conference Computer Vision (ICCV), pp. 563-570, November 2011
12. Roopak, M., Yun Tian, G., and Chambers, J.: "Deep learning models for cyber security in IoT networks", 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 452-457, 2019
13. LeCun, Y., and Bengio, Y.: "Convolutional networks for images, speech, and time series", in M. A. Arbib (eds.), The handbook of brain theory and neural networks, pp. 255-258, 1998
14. Osia, S. A. , Shamsabadi, A. S., Taheri, A., Rabiee, H. R., and Haddadi, H.: "Private and scalable personal data analytics using hybrid Edge-to-Cloud deep learning", Computer, vol. 51, no. 5, pp. 42-49, May 2018