International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
http://ijcjournal.org/

Peer-to-peer Approach for Distributed Privacy-preserving Deep Learning

Mustapha Abdulkadir Sani[a], Abdulmalik A. Lawan[b*], Salisu Mamman Abdulrahman[c]

[a,b,c]*Department of Computer Science, Kano University of Science and Technology, Wudil 713281, Nigeria*
[a]*Email: mustapha.abdulkadir2205@gmail.com*
[b]*Email: aalawan@kustwudil.edu.ng*
[c]*Email: salisu.abdul@gmail.com*

Abstract

The revolutionary advances in machine learning and Artificial Intelligence have enables people to rethink how we integrate information, analyze data, and use the resulting insights to improve decision making. Deep learning is the most effective, supervised, time and cost efficient machine learning approach which is becoming popular in building today's applications such as self-driving cars, medical diagnosis systems, automatic speech recognition, machine translation, text-to-speech conversion and many others. On the other hand the success of deep learning among others depends on large volume of data available for training the model. Depending on the domain of application, the data needed for training the model may contain sensitive and private information whose privacy needs to be preserved. One of the challenges that need to be address in deep learning is how to ensure that the privacy of training data is preserved without sacrificing the accuracy of the model. In this work, we propose, design and implement a decentralized deep learning system using peer-to-peer architecture that enables multiple data owners to jointly train deep learning models without disclosing their training data to one another and at the same time benefit from each other's dataset through exchanging model parameters during the training. We implemented our approach using two popular deep learning frameworks namely Keras and TensorFlow. We evaluated our approach on two popular datasets in deep learning community namely MNIST and Fashion-MNIST datasets. Using our approach, we were able to train models whose accuracy is relatively close to models trained under privacy-violating setting, while at the same time preserving the privacy of the training data.

Keywords: Data privacy; deep learning; deep learning models; distributed systems.

---

* Corresponding author.

1. Introduction

Deep learning is among the recent most popular techniques of machine learning used in building today's applications like self-driving cars [1], diagnoses of diseases [2], speech recognition [3], text-to-speech conversion [4], machine translation [5] etc. Deep learning models are normally built using computing systems called Artificial Neural Networks (ANN) that are inspired from biological neural networks and are designed to recognize useful representations from data and then use the representations to make a prediction about unseen data [6]. Recently, algorithmic breakthroughs, availability of large data and increased computational power have significantly contributed in building deep learning models that reached near-human-level performance [7]. However, doing deep learning is associated with a number of challenges, among which is preserving the privacy and confidentiality of the training data. Depending on the domain of application, privacy-preserving policies usually limits accessibility to large-scale training data that are essential in building generalizable deep learning models. For instance, clinical studies are usually limited by the inaccessibility to large-scale clinical records. Thus, deep learning models built based on such sensitive and limited data will be less accurate and none generalizable [8], [9]. Recently, unlike the traditional centralized deep learning approaches, studies have suggested decentralized deep learning protocols that could preserve the privacy of the data used for training deep learning models. Most notably, [9] demonstrated the efficacy of collaborative server-based architecture in the preserving the privacy of training data for the deep learning models. However, the collaborative architectures are not fully decentralized; if the parameter server fails during the training, the entire system collapse. The present study proposed a fully decentralized deep learning protocol based on peer-to-peer exchange of model parameters, which preserves the privacy of the training data while substantially maintaining the models' accuracy. Consequently, model training was conducted under two different scenarios namely privacy-violating scenario and privacy-preserving scenario to provide comparative findings on the models' accuracies and to investigate the efficiency of the averaging techniques applied on the model parameters. The remainder of this paper is organized as follows. In Section 2, we present an overview of existing work in related areas. Section 3 describes our proposed peer-to-peer distributed approach as well the datasets used in our experiments. Section 4 provides details about the experiments and results. The final section presents conclusions and future work.

2. Related work

Traditionally, deep neural networks are trained in a centralized manner in which all dataset needed for training the model is uploaded to a central location. This kind of approach is very effective in terms of producing accurate models, but the privacy of training data is not preserved [7]. In order to address this privacy issue, studies proposed a distributed collaborative deep learning approach in which models are trained locally and only a fraction of the training parameters are exchanged [7,9–12]. This approach yields models that are as accurate as the individual models built using centralized approach, and at the same time preserving the privacy of participant's training data [7]. Thus, in the centralized, privacy-violating deep learning approach service provider uploads all participants' training data to a central location for training. While, in the collaborative, privacy-preserving scenario participants train their models in distributed manner while exchanging their models parameters. Both centralized and distributed scenarios are further explained with the help of Figure 1 and Figure

2, respectively. There are several studies on collaborative deep learning [9–11,13]. However, the work of [9] is much related to the present study. [9] proposed a distributed deep learning protocol to demonstrated how the participants' privacy could be preserved without sacrificing the models' accuracy. In their approach, identical learning objective and neural network architecture was adopted, while multiple participants preserved his training data locally. In their deep learning protocol, participants train their models in collaborative manner by sharing among themselves a fraction of parameters of their individual models. Communication between the participants is done through what they called a parameter server. During each training epoch, each participant is going to upload fraction of his model parameters to the parameter server and download the updated parameters. Their approach was able to yield model, whose accuracy is close to model trained under privacy-violating setting. One major drawback of their approach is that their deep learning system is not fully decentralized; there is still some sort of centralization by relying on parameter server to upload and download model parameters. Consequently, if the parameter server fails during the training, the entire system is going to collapse. Therefore, in order to avoid having a single point of failure the present study proposed the use of a fully decentralized training system based on peer-to-peer architecture.

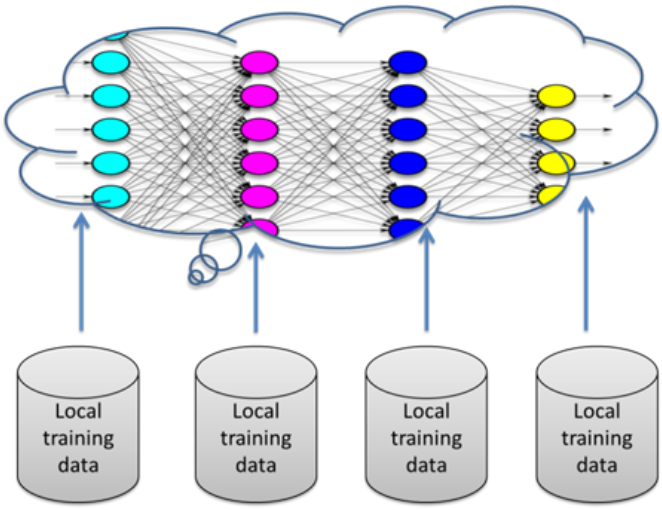

Figure 1: Centralized Deep Learning, a privacy violating-scenario

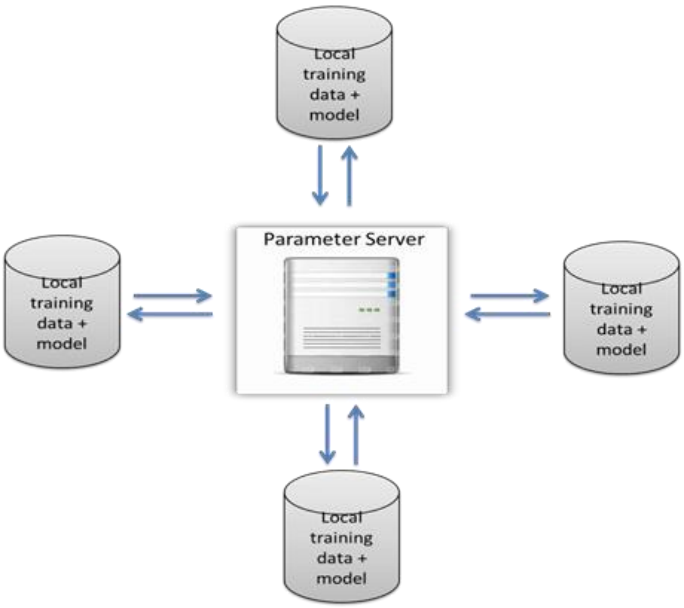

Figure 2: Collaborative Deep Learning, a privacy-preserving scenario

3. Method

3.1 The proposed peer-to-peer distributed approach

The proposed system architecture is illustrated with the help of Figure 3. The detailed description of the proposed distributed deep learning protocol of the study assumed four participants; each of which has a local dataset for model training. Specifically, each of the two datasets (i.e. MNIST and Fashion-MNIST datasets) we experimented on contains 60,000 training samples and 10,000 test samples. Hence, we split the training set in each dataset into four and allocate 15,000 samples to each participant. The details about the MNIST and Fashion-MNIST datasets will follow shortly. Accordingly, each participant trained his model using the 15,000 samples allocated to him and evaluated the model with the allotted 10,000 test samples. Noteworthy, it is assumed that all the participants agreed on using identical neural network architecture and learning objective. Furthermore, we used peer-to-peer communication architecture in which all the participants are fully connected to each other. The reason behind making this choice is to avoid having a single point of failure. Therefore, using our communication architecture, even if one or two peers collapsed the remaining peers will continue the training and benefit from one another. In summary, our approach enables each of the participants to build and train his model locally by iterating over many epochs. However, during the training all participants exchange with one another some parts of their model parameters in asynchronous manner. Once the training is done each participant will evaluate the model independently without any collaboration with other participants. The pseudo code of our distributed algorithm is depicted in Figure 4.

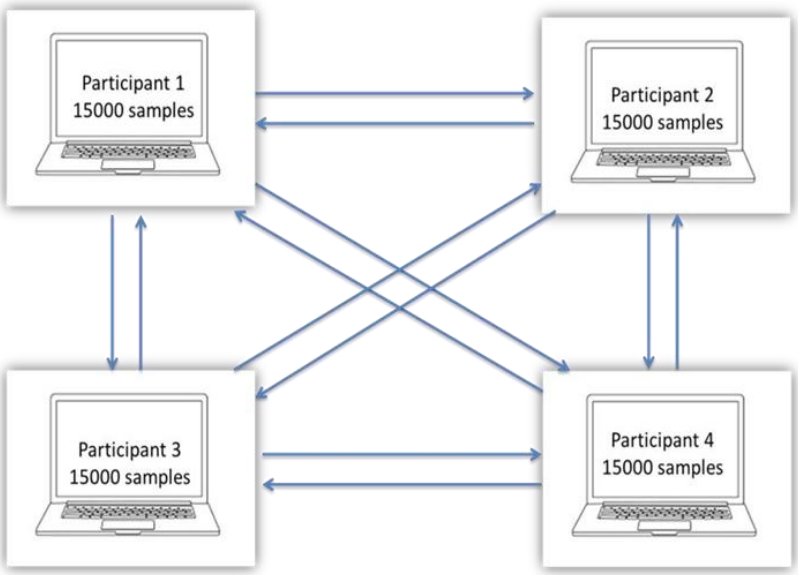

Figure 3: The proposed distributed deep learning protocol architecture using peer-to-peer architecture

Procedure Training(Participants P=(p_1, p_2, p_3, p_4), Epochs E=(e_1, e_2,…e_n))
For p_i, i ∈{1,2,3,4} do
- Build the model
- Compile the model
- For e_j, j ∈{1,2,3…,n} do
  - Extract the model weights
  - Select weight values which are above a given threshold
  - Send the selected weights to all other participants
  - On the event of receiving the weights from other participants
  - Take the average of your weights and the received weights
  - Set the averaged weights to the model
- Evaluate the model

Figure 4: Pseudocode of our distributed algorithm

We implemented this algorithm using Message Passing Interface (MPI). We created four processes to simulate the four participants. When the algorithm is running, the processes will be communicating by exchanging messages, the messages exchanged are model weights that are represented as numpy arrays. All our codes are available at [14].

3.2 Datasets used in the study

We evaluate our approach on two popular datasets in the machine learning community namely; MNIST and Fashion-MNIST datasets[15,16].

3.2.1 MNIST dataset

MNIST dataset [15] is a collection of gray scale images of handwritten digits ranging from 0 to 9 with their

respective labels. Each image is a grid of 28 by 28 pixels. The dataset consists of 60,000 training samples and 10,000 test samples. Figure 5 shows some examples of images of handwritten digits in the MNIST dataset.

Figure 5: Sample images in MNIST dataset

3.2.2 Fashion-MNIST dataset

Fashion-MNIST dataset [16] is a collection of gray scale images of clothing materials ranging from shirts, sandals, sneakers, trousers, bags, ankle boots, coats and t-shirts. Like MNIST dataset, it also consists of 60,000 training samples and 10,000 test samples. Figure 6 shows some examples of the samples contain in the dataset.

Figure 6: Sample images in Fashion-MNIST dataset

The learning objective in case of MNIST dataset is to classify the input as one of the 10 possible digits, and therefore, the size of the output layer is 10. In the case of Fashion-MNIST the learning objective is to classify the input as the one of the 10 possible clothing materials, the size of the output layer is also 10. Table 1 depicted the sizes of the two datasets considered in the study.

Table 1: Size of training and test datasets

	MNIST	Fashion-MNIST
Train set	60,000	60,000
Test set	10,000	10,000

3.3 Computing framework employed in the study

We used Keras and TensorFlow for building our deep neural networks. Keras [https://keras.io/] is a high-level API for building deep neural networks, which is written in Python and capable of running on top of either TensorFlow [https://www.tensorflow.org] (a deep learning framework developed by Google), Theano [http://deeplearning.net/software/theano] (also a deep learning framework developed by the MILA lab at Université de Montréal) or CNTK [https://github.com/Microsoft/CNTK] (which is also a deep learning framework developed by Microsoft). Keras was developed to enable researchers to conduct fast experimentation and it does not handle low-level operations such as tensor manipulation and differentiation. Instead, it relies on a specialized, well-optimized deep learning frameworks mentioned above to serve as the backend engine to it [17]. In our work, we used TensorFlow as the backend engine.

4. Experiments, Results and Discussions

4.1 Results of centralized training

In the centralized training scenario, all the 60,000 training samples in each of the two datasets were pooled into a single location and the models were trained on the entire 60,000 training samples and tested on the 10,000 test samples in each case. This scenario is privacy-violating scenario because the trainer has access to all training data of other participants. After testing the models on the 10,000 test samples; we were able to achieve 99.23% and 92.40% accuracy on MNIST and Fashion-MNIST respectively. These results as depicted in Table 2 are expected because the models had access to all the training data.

Table 2: Result of the centralized training

	Trainer (service provider)
Test Accuracy on MNIST	99.23%
Test Accuracy on Fashion-MNIST	92.40%

4.2 Results of standalone training

In this scenario we split each of the two datasets into 4 and allocated 15,000 samples to each participant, each participant solely train his models on the 15,000 samples allocated to him from each dataset without any collaboration. This scenario is privacy-preserving scenario since participants did not exchange any information. After training the models in this setting on the two datasets, each participant tested his models on the same 10,000 test samples from each datasets and each participant was able to achieve on average accuracy of 98.42% and 89.90% on MNIST and Fashion-MNIST respectively. These results are presented in the Table 3.

Table 3: Results of standalone scenario

	Test Accuracy on MNIST	Test Accuracy on Fashion-MNIST
Participant 1	98.49%	90.32%
Participant 2	98.53%	89.83%
Participant 3	98.49%	89.60%
Participant 4	98.15%	89.84%
Average	98.42%	89.90%

4.3 Results of collaborative training "share all weights"

In this scenario, we split each of the two datasets into four and allocated 15,000 samples to each participant, each participant trains his models using the 15,000 training samples allocated to him from each dataset. This scenario is also privacy-violating scenario because during the training all participants are going exchange all their model weights with one another. If it happens among the participants there is a malicious user he may try to infer some confidential information from other participants training data if he has access to all their model weights. After training the models in this setting, each participant tested his models on the same 10,000 test examples from each of the two datasets and on average each participant was able to achieve on average accuracy of 98.85% and 90.76% on MNIST and Fashion-MNIST respectively. These results are presented in the Table 4 below.

Table 4: Results of "share all weights" scenario

	Test Accuracy on MNIST	Test Accuracy on Fashion-MNIST
Participant 1	98.85%	90.74%
Participant 2	98.82%	90.63%
Participant 3	98.85%	90.93%
Participant 4	98.87%	90.74%
Average	98.85%	90.76%

4.4 Results of collaborative training MNIST "share part of weights"

In order to avoid attack on privacy of the participant's training data, in our approach, instead of participants exchanging all their weights, we split the weights into two parts namely, private weights and public weights. During the training each participant is going to share asynchronously with other participants only the public weights. By doing this even if the malicious user intends to infer something he will not succeed because the information he has is in-complete. There are so many criteria that can be used to split the weights into private and public. One approach is to sort the weights and select the largest values. Another approach is to select weight values that are above or below a particular threshold value. In our case we use the latter approach.

During each training epoch each participants will select weights values that are above a given threshold value and share those weights with other participants in asynchronous manner. We set this threshold value to -0.25 and 0.25 in the case of experiments on MNIST and Fashion-MNIST respectively. At the end of the training each participant tested the models he built using each of the two datasets on the 10,000 training samples. Each participant was able to achieve on average accuracy of 98.71% and 90.11% on MNIST and Fashion-MNIST respectively. These results are presented in the Table 5 below.

Table 5: Result of "share part of weights scenario"

	Test Accuracy on MNIST	Test Accuracy on Fashion-MNIST
Participant 1	98.72%	90.14%
Participant 2	98.74%	90.12%
Participant 3	98.68%	90.25%
Participant 4	98.71%	89.93%
Average	98.71%	90.11%

As can be seen from the results we obtained on the experiments on MNIST dataset, using the approach we proposed "Share Part of the Weights" participants were able to achieve on average accuracy of **98.71%,** slightly above the "Standalone" training in which participants obtained on average accuracy of **98.42%** and very close to privacy-violating setting "Share all Weights" where participants achieved on average accuracy of **98.85%.** Using our approach we are still able to get results relatively very close to privacy-violating setting while at the same preserving the privacy of participant's training data. These results are summarized in Table 6.

4.5 Results of experiments on Fashion-MNIST dataset

In the case of the experiments on Fashion-MNIST dataset, using the approach we proposed "Share Part of the Weights", participants were able to achieve on average accuracy of **90.11%,** slightly above the "Standalone" training in which participants obtained on average accuracy of **89.90%** and very close to privacy-violating setting "Share all Weights" where participants achieved on average accuracy of **90.76%.** Similarly, in this case, using our approach we are still able to get results relatively close to privacy-violating setting while at the same preserving the privacy of participant's training data. These results are summarized in Table 6.

Table 6: Overall test accuracy averaged over the four participants on the two datasets

Scenario	Privacy	Test Accuracy on MNIST	Test Accuracy on Fashion-MNIST
Centralized	Privacy-violated	99.29%	92.40%
Share All Weights	Privacy-violated	98.85%	90.76%
Standalone	Privacy-preserved	98.42%	89.90%
Share Part of the Weights	Privacy-preserved	98.71%	90.11%

The following plots (i.e. Figure 7-15) show the training and validation accuracy on the two datasets under different settings, x-axis is the number of times the entire training data is passed into the network i.e. epoch, and y-axis corresponds to the accuracy achieved during training and validation.

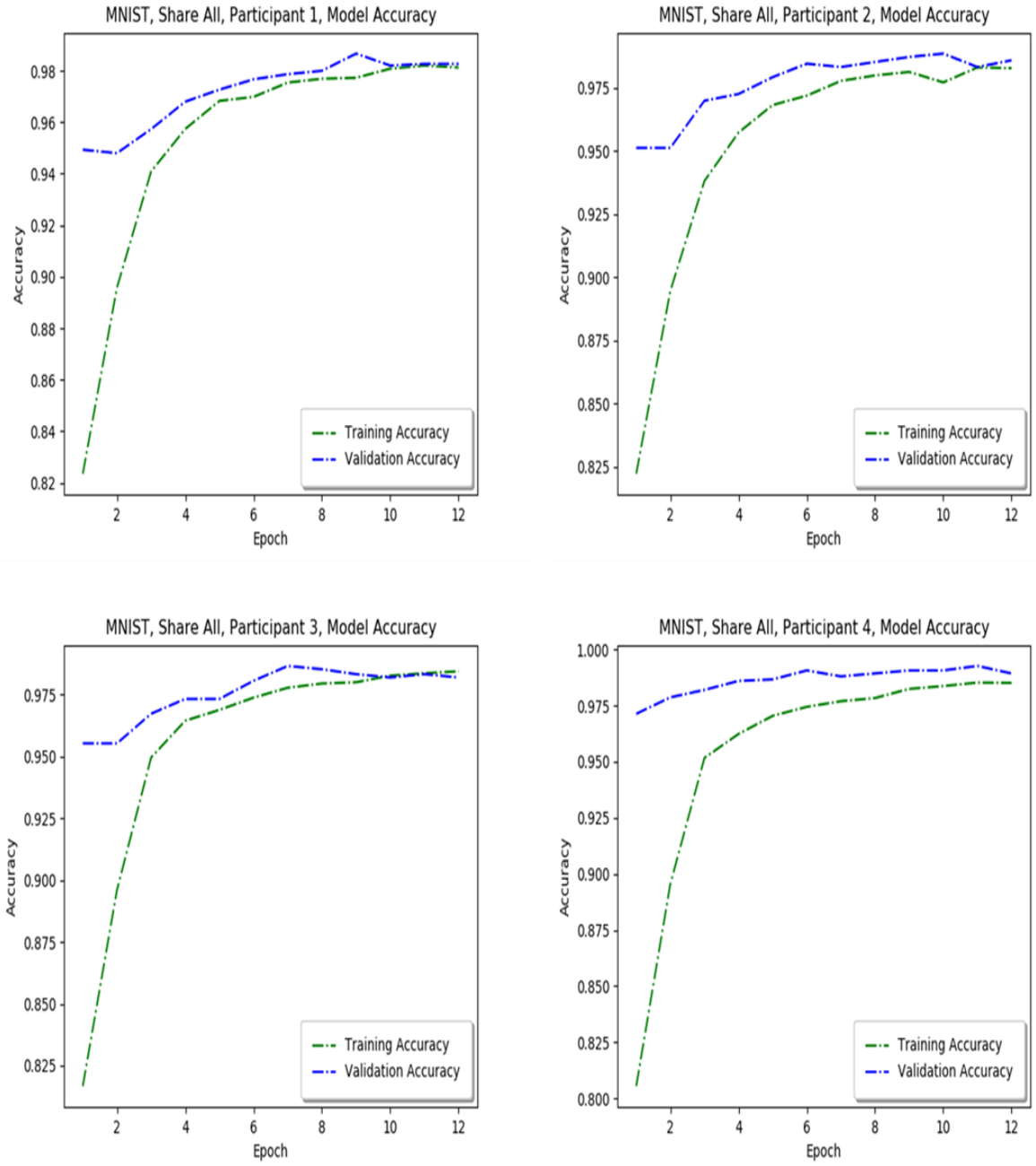

Figure 7: Training and Validation accuracy for MNIST "share all" collaborative scenario

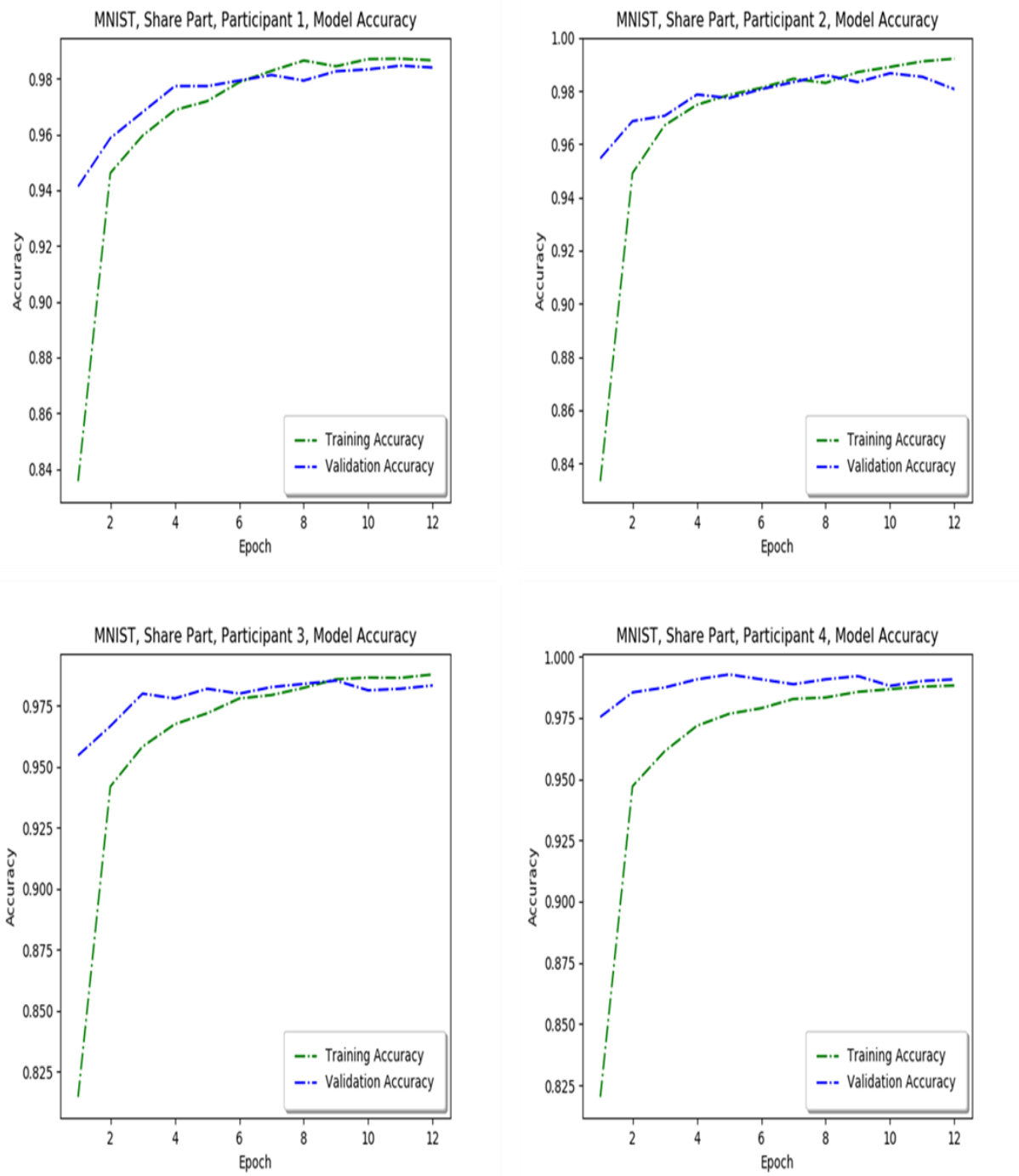

Figure 8: Training and Validation accuracy for MNIST "share part" collaborative scenario

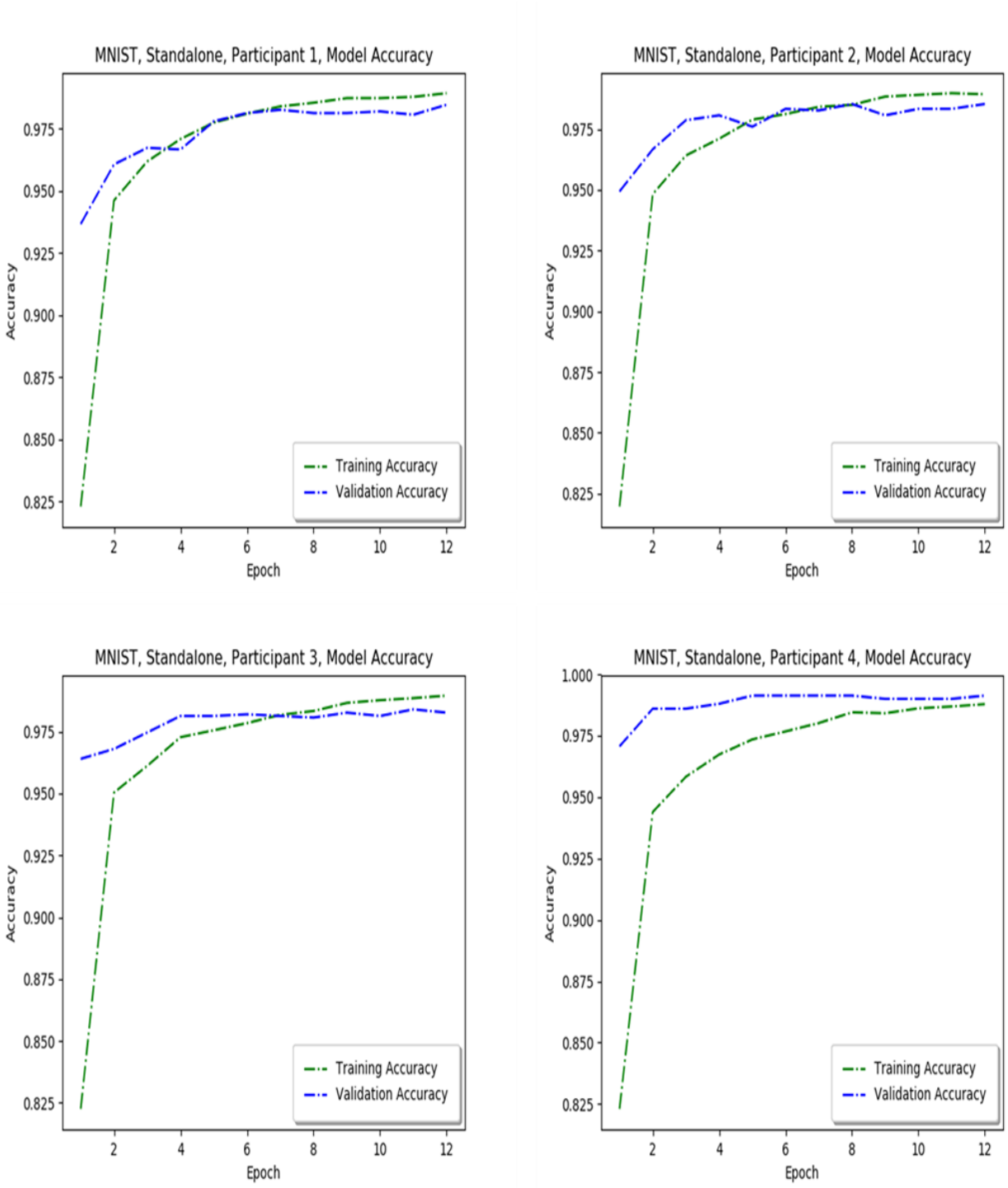

Figure 9: Training and Validation accuracy for MNIST standalone scenario

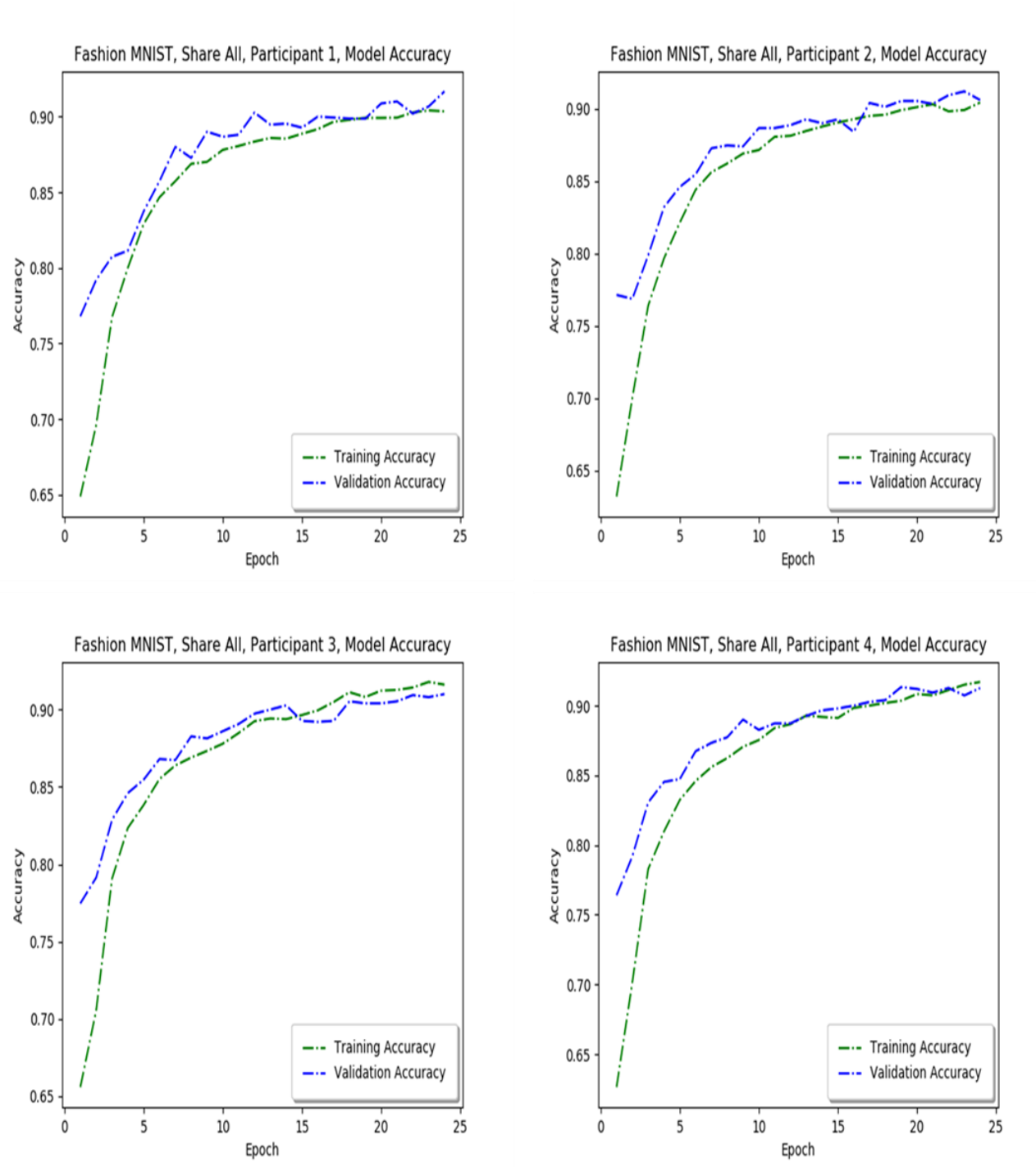

Figure 10: Training and Validation accuracy for Fashion MNIST collaborative scenario

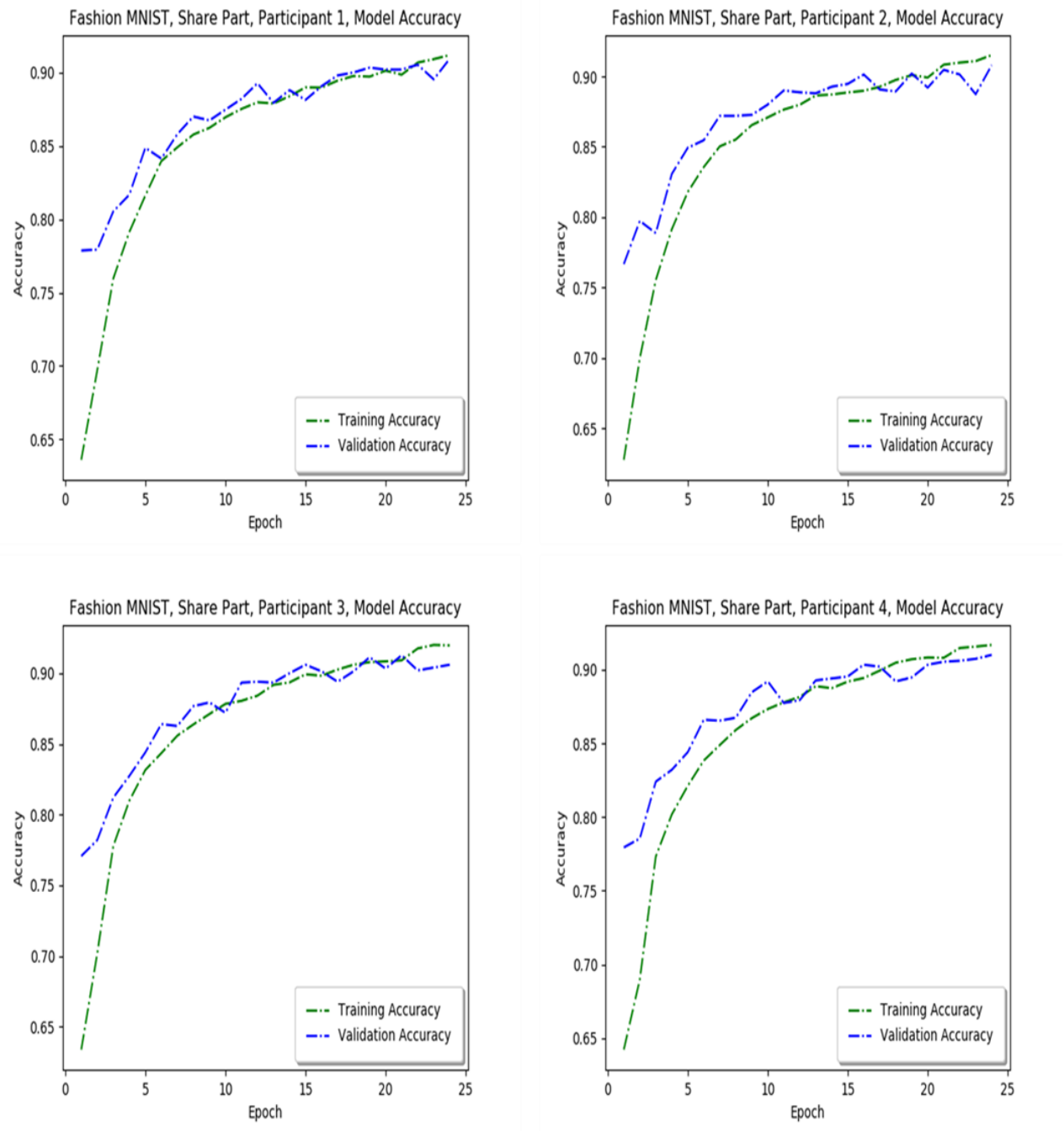

Figure 11: Training and Validation accuracy for "share part" Fashion MNIST collaborative scenario

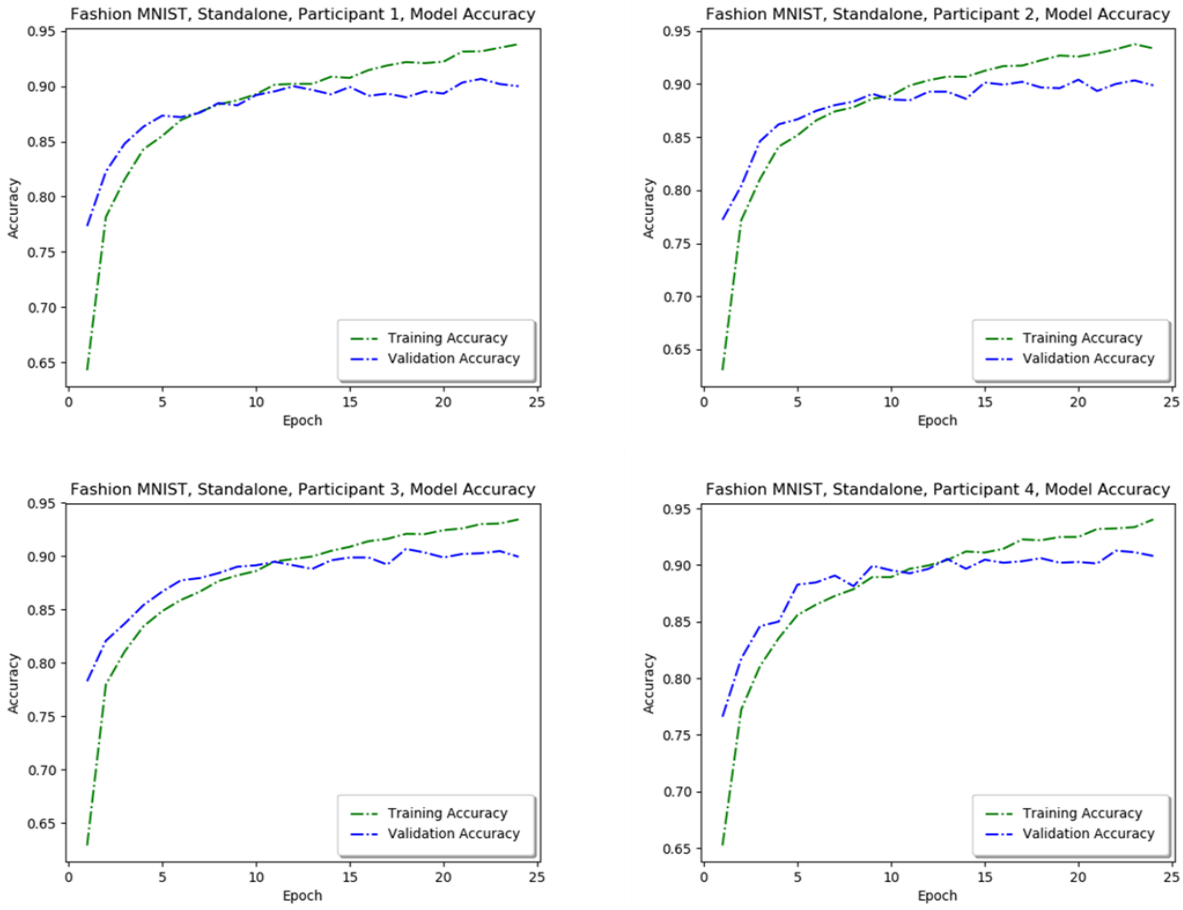

Figure 12: Training and Validation accuracy for Fashion-MNIST standalone scenario

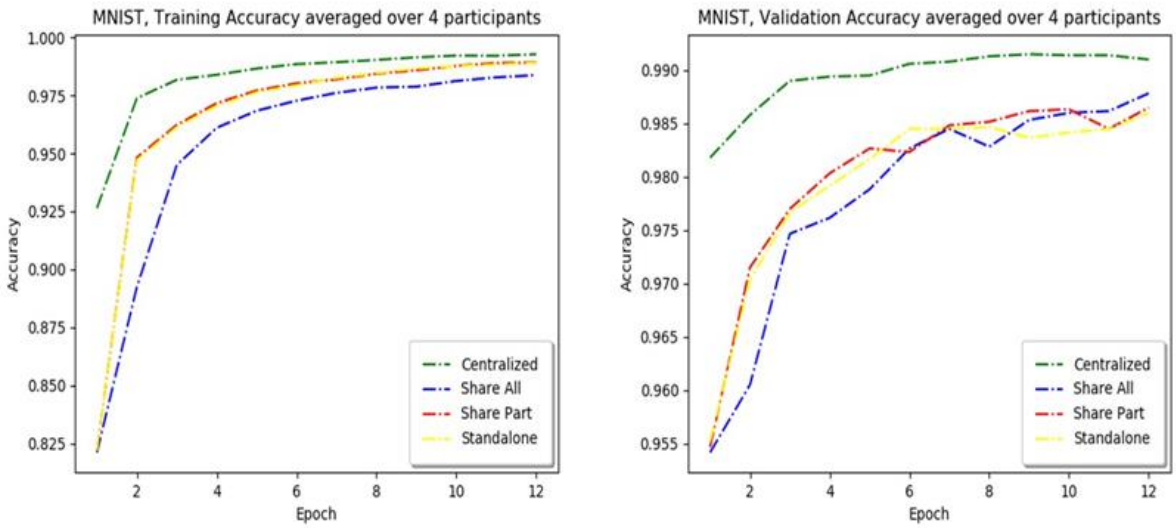

Figure 13: Training accuracy (left plot) and Validation accuracy (right plot) on MNIST dataset

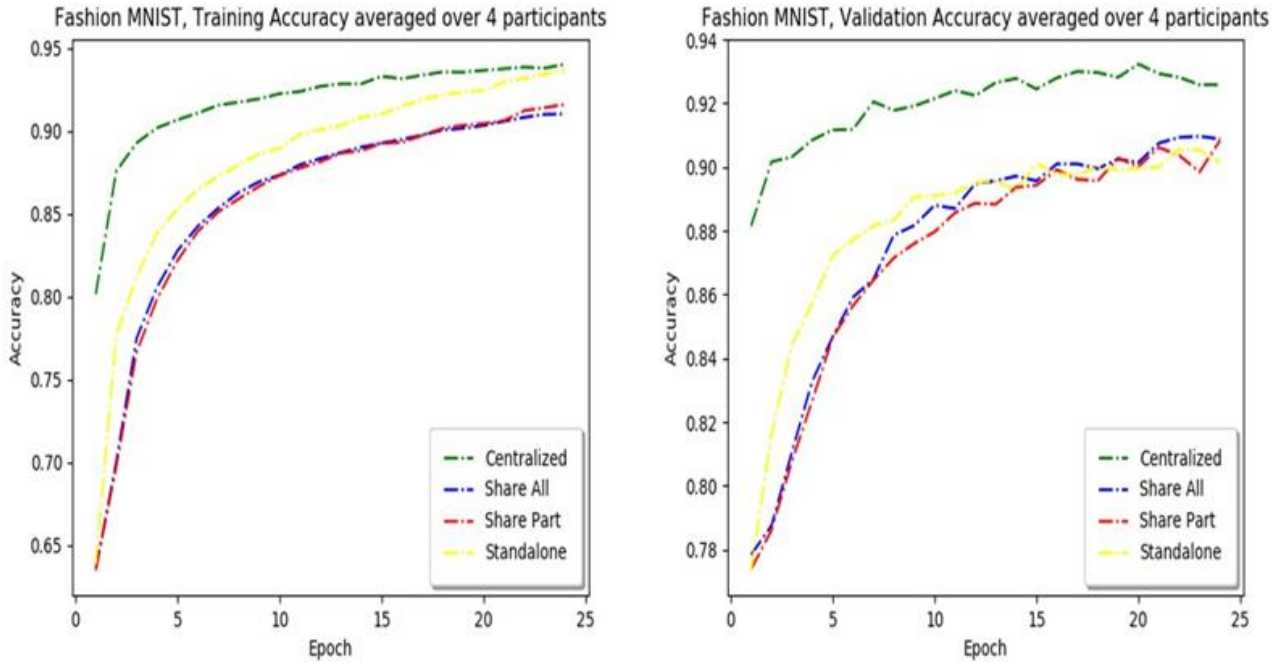

Figure 14: Training accuracy (left plot) and Validation accuracy (right plot) on Fashion-MNIST dataset

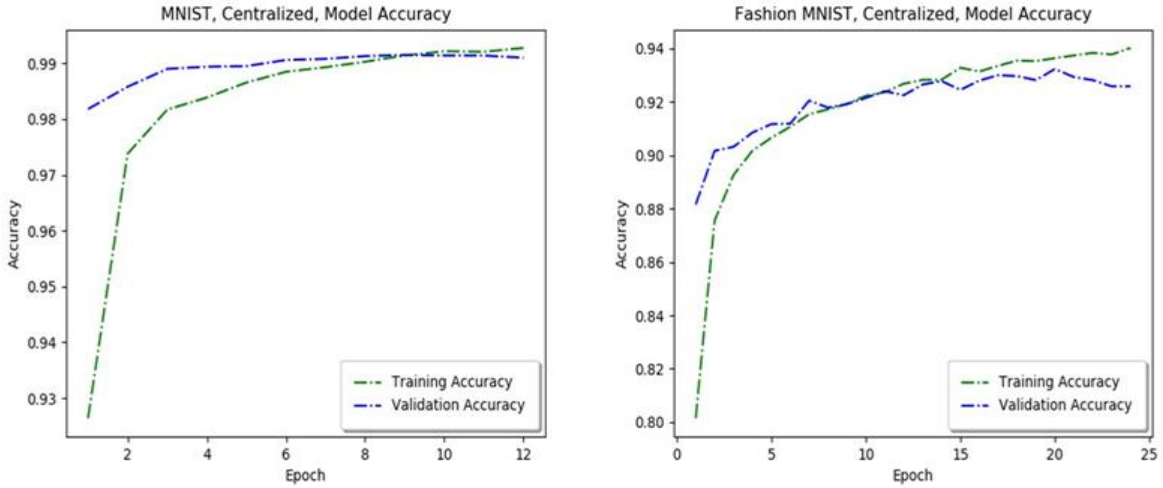

Figure 15: Training and Validation accuracy when the model is trained in a centralized manner

5. Conclusions and Recommendations

5.1 Conclusions

In this work, we proposed a decentralized deep leaning system based on exchanging and averaging model parameters using peer-to-peer architecture. Our methodology enables data owners to train deep learning models in a collaborative manner without disclosing their training data to one another, thus preserving the privacy and confidentiality of their training data. Using our approach, we are able to get results close to privacy-violating

scenario. Our findings show that exchanging and averaging part of model parameters between participants results in little improvement in model accuracy compared to a situation where participants train their model without collaboration. In conclusion, our approach can bring benefits of deep learning to domains where privacy and confidentiality of training data need to be preserved.

5.2 Recommendations

In the process of carrying out this research work, we encountered two major challenges. The first is the lack of access to real medical dataset. The second challenge is unavailability of powerful hardware needed for efficient experimentation. Future studies should explore the use of real medical dataset and more powerful machine configurations instead of MNIST and Fashion MNIST on cluster of machines.

6. Declaration of competing interest

The authors declare that they have no conflict of interest

References

[1]. A. Uçar, Y. Demir, and C. Güzeliş, "Object recognition and detection with deep learning for autonomous driving applications," Simulation, vol. 93, no. 9, pp. 759–769, Sep. 2017.

[2]. D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep Learning for Identifying Metastatic Breast Cancer," pp. 1–6, 2016.

[3]. A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, vol. 3, no. 3, pp. 6645–6649.

[4]. S. Arik et al., "Deep voice: Real-time neural text-to-speech," 34th Int. Conf. Mach. Learn. ICML, vol. 1, no. Icml, pp. 264–273, 2017.

[5]. L. Deng and Y. Liu, "Deep Learning in Machine Translation," in Deep Learning in Natural Language Processing, L. Deng and Y. Liu, Eds. Singapore: Springer Singapore, 2018, pp. 1–327.

[6]. F. Chollet, "What is deep learning," in Deep Learning with Python, NY, USA: Manning Publications Co., 2018, pp. 3–24.

[7]. B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 603–618.

[8]. M. Al-Rubaie and J. M. Chang, "Privacy preserving machine learning: Threats and solutions," 2018.

[9]. R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," 2015 53rd Annu. Allert. Conf. Commun. Control. Comput. Allert. 2015, pp. 909–910, 2016.

[10]. B. K. Beaulieu-Jones, S. G. Finlayson, W. Yuan, and Z. S. Wu, "Privacy-preserving distributed deep learning for clinical data," 2018.

[11]. Y. Wang et al., "Privacy Preserving Distributed Deep Learning and Its Application in Credit Card Fraud Detection," in 2018 17th IEEE International Conference On Trust, Security And Privacy In

Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018, no. August, pp. 1070–1078.

[12]. M. K. Abd-ellah, A. Ismail, A. A. M. Khalaf, and H. F. A. Hamed, "A review on brain tumor diagnosis from MRI images : Practical implications , key achievements , and lessons learned," Magn. Reson. Imaging, vol. 61, no. August 2018, pp. 300–318, 2019.

[13]. A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," Int. Conf. Artif. Intell. Stat. AISTATS 2018, pp. 473–481, 2018.

[14]. M. Abdulkadir, "Distributed Privacy-Preserving Deep Learning," GitHub, 2019. [Online]. Available: https://github.com/MustaphaAbdulkadir1983/distributed_deep_learning. [Accessed: 15-Feb-2021].

[15]. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[16]. H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mniST: A novel image dataset for benchmarking machine learning algorithms," 25-Aug-2017. [Online]. Available: https://github.com/zalandoresearch/fashion-mnist. [Accessed: 15-Feb-2021].

[17]. F. Chollet, "Introduction to Keras," in Deep Learning with Python, NY, USA: Manning Publications Co., 2018, pp. 61–62.