Final Year Project Literature Survey Report on

"Multi-Task Federated Learning for Personalised Deep Neural Networks"

Submitted in partial fulfilment for the requirement of the degree of

Bachelor of Technology (B.Tech)

In

Computer Science and Engineering,

Artificial Intelligence and Machine Learning

By

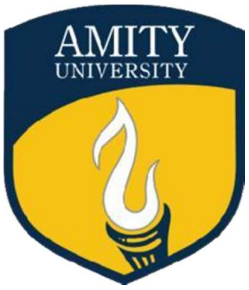
Mr. Sadi Gautam Reddy (A70405219029)

Ms. Pratima Chinta (A704132519007)

Ms. Sanjana Paninjayath (A70405219005)

Course In charge

Dr. Satheesh Abimannan



Department of Computer Science and Engineering

Amity School of Engineering and Technology
Amity University Maharashtra.

2022-2023

Declaration

I declare that this report represents my ideas in my own words and where others ideas are included, I have referenced the original sources.

I have adhered to all principles of academic honesty,integrity and have not falsified any fact in this report. I understand that any violation of the above will cause disciplinary action by the institute and can evoke penal action from sources which have not been properly cited or whose permission has not been taken when needed.

Pratima Chinta (A704132519007)

Sadi Gautam Reddy (A70405219029)

Sanjana Paninjayath (A70405219005)

Place: Amity University Mumbai

Date:31 /12 /2022

CERTIFICATE

This is to certify that the research paper report entitled "**Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing**" is a bonafide work of **Mr Sadi Gautam Reddy**, **Ms Pratima Chinta** and **Ms**. **Sanjana Paninjayath** submitted to **Amity University Maharashtra** in fulfillment of the requirement for the Final Year Project Literature Survey for the semester **VII.**

Mr. Chaitanya V Mahamuni
Course In charge

Dr. Deepa Parasar
Department in-charge
(Computer Science and Engineering)

Place:

Date:31 /12 /2022

Acknowledgment

I would like to express my heartfelt gratitude towards my supervisor Dr. Satheesh Abimannan Sir, for his continuous guidance for my academic project, for his motivation, enthusiasm and great knowledge. I want to thank him for encouraging me to do the project in the field of Edge computing, Neural networks. His dynamism, presence and optimism have provided an invaluable influence on my career and outlook for the future. I consider it my good fortune to have got an opportunity to work with such a brilliant person. I express my gratitude to Dr. Deepa Parasar, Department in-charge, Department of Computer Science and Engineering, for extending all possible help throughout the work directly or indirectly. They have been a great source of inspiration to me and I am immensely obliged. I would like to acknowledge my institute, Amity School of Engineering and Technology, for providing good facilities to complete my project work. I am especially thankful to my parents because this project would not have been complete without their support, love and elevating inspiration. They are my teachers after I came to this world and have set a great example for me about how to live, study, and work.

ABSTRACT

Federated Learning (FL) is a new approach that trains Deep Neural Networks (DNNs) on IOT devices, by keeping the private user data on the clients and not allowing the data to leave the devices. The convergence speed of the FL algorithms are harmed by non-Independent and Identically Distributed (non-IID) user data. Most of the existing works on Federated Learning measures global-model accuracy, but in many cases, such as user content-recommendation, improving the individual User model Accuracy (UA) is the real objective. Hence, we propose a Multi-Task Federated Learning (MTFL) algorithm that introduces non-federated Batch-Normalization (BN) layers into the federated Deep Neural Networks. Convergence speed and UA are strengthened by MTFL by allowing users to train models personalised to their own data. Iterative FL optimization algorithms such as Federated Averaging (FedAvg) are adaptable with MTFL. Experiments using MNIST and CIFAR10 datasets demonstrate that MTFL is able to significantly reduce the number of rounds by up to 5x, which are required to reach a target UA, with a further improvement in performance when using FedAvg-Adam.

INTRODUCTION

Multi access edge computing moves cloud applications to edge thereby enabling low-latency and real-time processing via content caching and computation offloading.MEC stores and processes huge quantities of data at the edge, close to their source.Deep Neural Networks (DNNs) for Machine Learning (ML) are becoming increasingly popular for their huge range of potential applications, state-of-the-art performance and ease of deployment. With the increasing size of DNN's, training them would become difficult because it is computationally expensive and would also require huge amounts of training data. Use of DNN's in MEC typically involves collecting data from IOT devices, doing the training on these models on cloud and finally deploying the model at the edge. Concerns about data privacy, however, has caused users to become unwilling to upload their sensitive data, raising the question about how these models will be trained.

FedAvg initializes a model at the server and then distributes this model to clients. The clients perform one round of training on their local datasets and then push their models to the server, which then averages these models and sends the aggregated model to the clients for successive training rounds. We refer to the entities that own the data for FL as ‘users’, and to the devices that participate in FL as ‘clients’.

PROBLEM STATEMENT AND OBJECTIVE:

We propose a MultiTask Federated Learning algorithm that introduces non-federated Batch normalization layers into federated DNN. Users train models personalized to their own data hence it improves User accuracy and convergence speed. Further, privacy is enhanced.

Iterative FL optimization algorithms such as Federated Averaging (FedAvg) are adaptable with MTFL.

CHALLENGES OF THE EXISTING METHOD

Federated Learning presents multiple unique challenges:

- Clients generally do not have Independent and Identically distributed (IID) data. The data is usually self-generated and may contain noise or may be just a subset of all the required features, which may cause hindrance in the training of the FL model.

- FL models typically use Global model accuracy as a performance metric on centralized test sets. However, the real objective is finding out the individual user model accuracy motivating personalized FL that would improve the local performance.

- Due to the non-IID nature of data,the performance of the global model may be better on some clients than others. Some clients may receive a worse model than the one they could have trained independently.

RELATED WORK/LITERATURE REVIEW:

As this work addresses several challenges to existing FL algorithms, we overview the related work in three sub- topics of FL: works considering personalisation, works dealing with practical and deployment challenges, and works aiming to improve convergence speed and global-model performance.

1.1 Personalised Federated Learning

Several writers have investigated 'personalising' FL models to adjust model performance to non-IID user datasets.

Meta-Learning tries to train a model that is simple to fine-tune with a small number of examples. Fallah et al. presented the Model Agnostic Meta-Learning (MAML)-based Per-FedAvg method, which adds a first-order adaptation term to the client loss functions, allowing them to be adapted to client datasets in a single step. Jiang et al. found a link between FedAvg and first-order MAML updates and developed a three-stage training approach to increase customisation. To boost personalization, several authors recommend training a mix of local and

global models in FL. Hanzely and Richt'arick included a learnable parameter that allows customers to vary the amount of local and global model mixing. Dinh et al. maintained a global model as well as a personal model for each user, running SGD on their personal model and then updating their copy of the global model in an outer loop. For the cross-silo FL context, Huang et al. preserved a local model on each client and added a proximal term to client loss functions to keep these models near to a 'personalised' cloud model.
Smith et al. introduced MOCHA, which executes Federated MTL and formulates FL as a function of a relationship matrix and a model weight matrix. Because their algorithm considers the varied hardware of clients, MOCHA is not directly comparable to our MTFL approach. Dinh et al. recently generalised MOCHA and other algorithms into the FedU framework, including a decentralised version.

To achieve customization in FL, we offer a Multi-Task learning technique (MTFL). We next demonstrate that our technique outperforms existing customised FL algorithms in terms of convergence time, personalisation performance, privacy, and storage cost.

1.2 Federated Learning in Edge Computing

At the network edge, FL executes distributed computing. The system design and communication costs of FL in this environment have been taken into account by several authors. An FL system was suggested by Jiang et al. that limits the amount of data that clients submit by choosing model weights with the biggest gradient magnitudes. Additionally, they took into account implementation specifics like asynchronous or round-robin client updates. A FedAvg system architecture was created by Bonawitz et al., with details on client/server roles, fault handling, and security. For their use of this technology with over 10 million clients, they also offer analytics. Duan et al. suggested the Astrea framework to handle the non-IID nature of client datasets in FL. Client datasets are enhanced to help lessen local class imbalances, and mediators are added to the global aggregation technique.
Additionally, some authors have looked into the effects of wireless FL clients. Yang et al. investigated several scheduling strategies in a wireless FL environment. According to their findings, simple FL schemes work well while the Signal-to-Interference-plus-Noise Ratio (SINR) is low, but as SINR rises, more sophisticated methods of client selection are required. A Hybrid Federated Distillation approach for FL with wireless edge devices was put forth by Ahn et al., and it included over-the-air computation and compression techniques. Their findings demonstrated that their system performed better in high-noise wireless environments.

The computational, networking, and communication resources of FL clients have been taken into account by other writers in edge computing systems. Wang et al. used smartphone studies to support their claim that FedAvg's calculation time—rather than its communication time—is the main bottleneck for real-world FL. They also proposed techniques to account for this computational heterogeneity. A technique created by Nishio and Yonetani reduces the amount of time needed to achieve a target accuracy for FedAvg by gathering data about the computer and wireless resources of clients before starting a round of FL.

These earlier studies suggested FL system implementations. MTL within FL, which is a major addition of our work with the MTFL algorithm, is not taken into account by them.

1.3 Federated Learning Performance

By distributing an initial model to participating clients, who each perform SGD on the model using their local data, the pioneering FedAvg algorithm cooperatively trains a model. A new round is started after these new models are sent to the server for averaging. There have been some improvements made to FedAvg's convergence rate. Leroy et al. updated the server's global model using Adam adaptive optimization. Additionally, Reddi et al. gave convergence guarantees and generalised additional adaptive optimization approaches in the same vein.. They differ from our FedAvg-Adam algorithm in that clients in FedAvg-Adam run Adam SGD (as opposed to vanilla SGD), and the Adam parameters and model weights are averaged on the server. As a different strategy for quickening convergence, Liu et al. applied momentum-SGD on clients and combined the momentum values of clients on the server with the model weights.

There have been some investigations into FL using client data that is non-IID or of poor quality. To reduce the disparities in the data distributions of the clients and increase the accuracy of the global models, Zhao et al. suggested exchanging a modest quantity of data amongst the clients. By determining the difference between a client model's local forecasts and predictions made using a reliable dataset, Konstantinov and Lampert determined which clients have low-quality data. By determining whether each client's update is consistent with the overall model during training, Wang et al. disregarded irrelevant client updates.

We further demonstrate that the FedAvg-Adam optimisation method provided here converges substantially more quickly than when using FedAvg or Adam optimisation just on the server because it leverages adaptive optimisation on clients rather than SGD.

PROPOSED METHOD

Through a federated learning process, multiple people can collaborate and train a deep learning model using the same data. Each party then downloads the model from the cloud, which is generally a pre-trained model. They then summarize and encrypt its new configuration after training it on their private data. Model updates are transferred back to the cloud, decrypted, averaged, and incorporated into the central model. Three types of distributed, decentralized training processes exist. With horizontal federated learning, similar datasets are used to train the central model. For example, movie and book reviews can be combined to predict someone's music preferences when vertical federated learning is used. The data are complementary in vertical federated learning; movie and book evaluations, for instance, are integrated to forecast a person's musical tastes. Finally, federated transfer learning involves training a foundation model that has already been trained to accomplish one task, such as detect automobiles, on a different dataset to perform another task, such as recognizing cats.

The MTFL algorithm is based on a client-server framework, but the rounds are initiated by the server. Here is the process followed:

1. First, the server will select all or a subset of all known clients from its database and ask them to participate in a FL round and will send them a Work Request message.
2. Clients will accept the work request depending on user preferences (for example, users can change their device's settings to participate in FL only when charging and connected to WiFi).
3. All receiving clients will send an acknowledgment message to the server.
4. The server will send the global model (and any associated optimization parameters) to all receiving clients, who will change their copy of the global model with private patches.
5. Next, Clients will perform local training using their own data to create a different model. Clients will store patch layers from their new model locally and upload their non-private model parameters to the server.
6. The server will wait for clients to finish training and upload their models. According to server preferences, it can either wait for a maximum timeout or for a certain number of clients to finish uploading..
7. Then the server will aggregate all received models and create one global model that is stored on the server before starting a new round.

The aim in FL is to minimise the objective function below:

$$F_{\mathrm{FL}} = \sum_{k=1}^{K} \frac{n_k}{n} \ell_k(\mathbf{\Omega}),$$

- total number of clients=K
- number of samples on client k is n_k
- n is the total number of samples across all clients,
- 'k' is the loss function on client k,
- V is the set of global model parameters.

Adding unique client patches to the FL model changes the objective function of MTFL to

$$F_{\mathrm{MTFL}} = \sum_{k=1}^{K} \frac{n_k}{n} \ell_k(\mathcal{M}_k) \qquad (2)$$

$$\mathcal{M}_k = (\Omega_1 \cdots \Omega_{i_1}, P_{k_1}, \Omega_{i_1+1} \cdots \Omega_{i_m}, P_{k_m}, \Omega_{i_m+1} \cdots \Omega_j), \qquad (3)$$

where Mk is the patched model on client k, composed of Federated model parameters Mk (j being the total number of Federated layers) and patch parameters Pk1 _ _ _ Pkm (m being the

total number of local patches, fig being the set of indexes of the patch parameters) unique to client k.

Algorithm 1. MTFL

1: Initialise global model Ω and global optimiser values V
2: **while** termination criteria not met **do**
3: Select round clients, $S_r \subset S, |S_r| = C \cdot |S|$
4: **for each** client $s_k \in S_r$ in parallel **do**
5: Download global parameters $\mathcal{M}_k \leftarrow \Omega$
6: Download optimiser values $V_k \leftarrow V$
7: **for** $i \in$ patchIdxs **do** ▷ Apply local patches
8: $\mathcal{M}_{k,i} \leftarrow P_{k,i}, V_{k,i} \leftarrow W_{k,i}$
9: **end for**
10: **for** batch b drawn from local data D_k **do**
11: $\mathcal{M}_k, V_k \leftarrow \text{LocalUpdate}(\mathcal{M}_k, V_k, b)$
12: **end for**
13: **for** $i \in$ patchIdxs **do** ▷ Save local patches
14: $P_{k,i} \leftarrow \mathcal{M}_{k,i}, W_{k,i} \leftarrow V_{k,i}$
15: **end for**
16: **for each** $i \notin$ patchIdxs **do**
17: Upload $\mathcal{M}_{k,i}, V_{k,i}$ to server
18: **end for**
19: **end for**
20: **for** $i \notin$ nonPatchIndexes **do**
21: $\Omega_i \leftarrow \text{GlobalModelUpdate}(\Omega_i, \{\mathcal{M}_{k,i}\}_{k \in S_r})$
22: $V_i \leftarrow \text{GlobalOptimUpdate}(V_i, \{V_{k,i}\}_{k \in S_r})$
23: **end for**
24: **end while**

Reference for the above image: Paper title: Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing, Authors: Jed Mills , Jia Hu , and Geyong Min

As shown in Algorithm 1, MTFL performs rounds of communication until certain exit criteria (such as the target UA) are met (line 2). At each round, a subset of clients Sr is selected to join from the set of all clients S (line 3). These clients save the global model V, which is a tuple of model parameters, and the global optimizer V, if used (lines 5-6). The client then updates its copy of the global model and optimizer with the private patch layer (lines 7-9) using the patchIdxs variable containing the index of the patch layer placement within the DNN. The customer trains using a copy of the currently personalized global model and the local data optimizer (line 10).

Depending on which FL optimization strategy is used within MTFL, the LocalUpdate function represents local training of the model. For FedAvg, LocalUpdate is just a mini-batch SGD. After local training, the updated local patch is saved (lines 11-13) and the non-patch layer and optimizer values are uploaded to the server (lines 14-16). The server follows his GlobalModelUpdate and GlobalOptimUpdate functions to create a new global model and a new

optimizer (lines 18-20 at the end of the round. These steps depend on the FL optimization strategy used. For example, FedAvg uses a weighted average of client models from GlobalModelUpdate. The updated global model marks the end of a round and a new round begins. The total complexity per round of MTFL is proportional to jSrj. where jSrj is the number of participating clients per round. The calculations performed by each client are independent of the total number of clients. MTFL (such as FedAvg) is highly scalable because the client performs local computations in parallel.

Scalability is needed in FL. This is because an actual deployment may have a large number of poorly performing clients. Updating the global model and optimizer (Algorithm 1, lines 20-23) depends on the optimization strategy used. For FedAvg and FedAvg-Adam, GlobalModelUpdate is basically a map-reduce (averaging after local training). For FedAdam, the Adam step after map reduction in GlobalOptimUpdate is independent of the number of clients. Choosing to use the BN layer for personalization within MTFL because 1) shows excellent personalization performance and 2) has a low storage cost for BN parameters (less than 1% of the total model size for the tested model architectures). Thus, all model layers can be kept private during MTFL, but there is a trade-off between the number of parameters kept private and the ability of the global model to converge.

SOFTWARE REQUIREMENTS

Package	**Version**
python	3.7
pytorch	1.7.0
torchvision	0.8.1
numpy	1.18.5
progressbar2	3.47.0

DATASETS

Requires MNIST and CIFAR10 datasets. For MNIST, the .gz files must be saved in a folder with path '../MNIST_data/', and for CIFAR10, the python pickle files must be saved in the folder '../CIFAR10_data/'.

PERFORMANCE METRICS-

We propose average User model Accuracy (UA) as a performance metric of FL. UA is the accuracy of a client using a local test-set. This test-set for each client should be drawn from a similar distribution as its training data. In this paper, we will perform experiments on

classification problems, but UA could be altered for different metrics (e.g., error, recall). In FL, user data is often non-IID, so users could be considered as having different but related learning tasks. It is possible for an FL scheme to achieve good global-model accuracy, but poor UA, as the aggregate model may perform poorly on some clients' datasets.

PREDICTED OUTCOME

Find the Effect of BN Patches on Inference i.e. to understand the impact that BN-patch layers have on UA.

Compare the personalisation performance of MTFL(FedAvg) with two other state-of-the-art Personalised FL algorithms: Per-FedAvg and pFedMe and FL (FedAvg) (where no model layers are private).

We will try to modify the algorithm and improve accuracy. We will try to minimize the latency of the algorithm.

REFERENCES:

https://www.ece.utexas.edu/events/federated-learning-and-analysis-multi-access-edge-computing

https://ieeexplore.ieee.org/document/9850408 -Decentralized Federated Averaging

T. Li, A. Sahu,V. Smith and A. Talwalkar, "Federated learning: Challenges, methods, and future directions," IEEE Signal Process. Mag., vol. 37, no. 3, pp. 50–60, May 2020.

Y. Jiang, J. Konecny, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," 2019, arXiv:1909.12488.

V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in Proc. Conf. Neural Inf. Process. Syst., 2017, pp. 4427–4437.

D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. Int. Conf. Learn. Representations, 2014.