Regularizing Federated Learning via Adversarial Model Perturbations

Nirbhay Sharma (B19CSE114)
Gautam Kumar (B19EE031)

May 9, 2023

Abstract

There are various recent attempts in Federated Learning (FL) to handle the data heterogeneity among the clients. Most of the approaches targeting data heterogeneity are based on homogeneous models. Due to their system homogeneous nature these approaches are based on parameter aggregation at the server end. There are various approaches recently proposed like FedAvg, FedProx etc. that targets FL settings. Most of these approaches try to perform some or other kind of regularizer at the client side. The regularizer ensures the more generalized training of client models so that when they would be aggregated at the server, the aggregated model perform well. In this project, we try to utilize the Adversarial Model Perturbations (AMP) regularizer to regularize clients' models. The AMP regulzaizer is based on perturbing the model parameters so as to get a more generalized model. The claim of AMP regularizer is to reach flat minima and therefore is expected to reach flat minima in FL settings as well. We analyze the effect of using AMP regularizer at the clients side. We perform experiments on publicly available datasets such as CIFAR10, CIFAR100 to analyze the effect. Our experiments compares various FL approaches before and after applying AMP regularizer. The code is available at Code

1 Introduction

Federated Learning (FL) is a privacy perserving machine learning paradigm which is primarily based on aggregation of various models received from clients. The most basic framework in this line of research is FedAvg [6]. The clients perform local training on their local datasets and communicate their locally trained models to the server. The server after receiving locally trained models aggregates their learned knowledege by directly averaging their model parameters. This initial algorithm is represented in fig. 1. After FedAvg, various papers comes under the same line of research. Since these algorithms are primarily based on aggregation. They don't perform much computation at the server end. They primarily try to improve clients performance. For improving clients performance the methods like SCAFFOLD [2], FedDyn [1], FedNTD [4], FedMLB [3], FedProx [5] comes up with various different kind of regularization scheme to regularize the clients models to make them more robust and high performing models.

Similarly the concept of adversarial perturbations found its application in many areas such as adversarial perturbations in terms of images leads to data augmentations which can be used to train neural networks to make them more robust to attacks etc. The paper AMP regularizer [7] talks about the similar application of adversarial perturbations at the model level. The paper claims to train a model adversarially by perturbing its parameters in the direction of gradient ascent. The perturbations in model parameters ensures the model to be regularize better and is highly likely to reach flat minima which in turn ensures a good performance. We, on the same line research try to analyze the effect of adversarial model perturbation training at the client side. The results of this analysis shows that for sure the performance is likely to improve if we perform AMP regularizer training at the client side in FL settings. Further we discuss the method, experiments and results for this project. Finally we conclude the project by providing conclusion in last section.

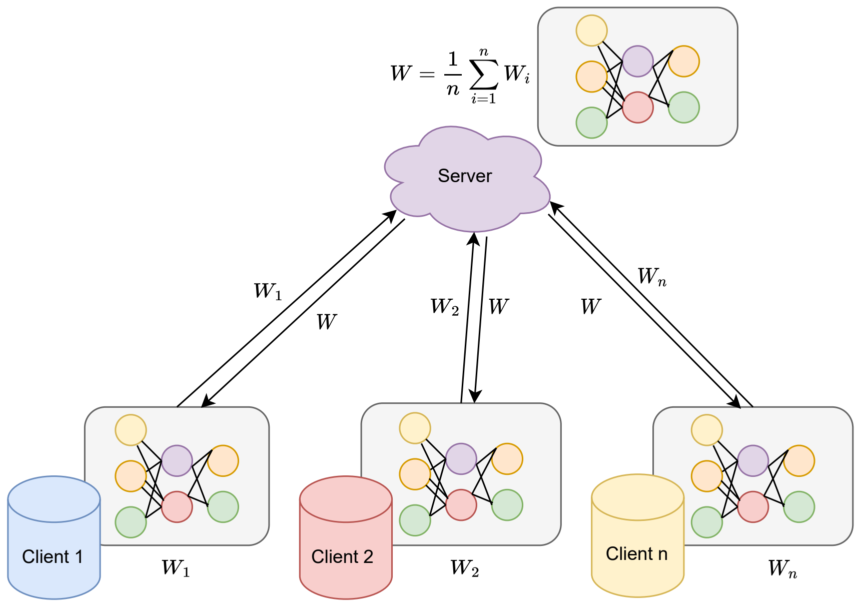


Figure 1: Representing basic FL framework. The clients train their models having weights W_i on their local database. They send the weights to the server. The server directly averages the weights to obtain a global model with weights W. The global weights are then send to each of the selected clients for further rounds of training.

2 Method

In this section we will discuss about the methodology followed to regularize clients using AMP regularizer.

2.1 AMP regularizer

Recent attempts in CVPR introduce AMP regularizer [7] which regularize neural networks by performing adversarial perturbations in their parameters. This ensures that network regularize better and is likely to reach flat minima. Therefore the authors proposes a regularization scheme which utilizes perturbations in model parameters to yield a more robust and generalized model which is robust to overfitting. The recent research shows the effectiveness of certain regularizer lies in its ability to reach flat minima. Thus flat minima becomes a crucial aspect in designing any regularizer techniques. To the same end, the authors proposes a novel loss function ($\mathcal{L}_{AMP}(\theta)$) which acts as a regularizer loss function. The core idea behind this loss function is to reach the flat minima. The significance of flat minima is such that, even if the test data has some slight difference in distribution than train data, due to flat minima it eventually reaches to minima thus flat minima helps in improving the robustness of the model. The empirical risk loss function which is naive loss function is prone to overfitting thus authors tries to make a robust empirical loss function by introducing model perturbations. The empirical loss function is as follows,

$$L_{EMP}(\theta) = \frac{1}{|D|}\sum_{i=1}^{|D|} l(x_i, y_i; \theta)$$

where D is the dataset and (x_i, y_i) are i^{th} datapoint in the dataset. and l is any general loss function for any task such as image classification. Then, authors tried to formulate the $L_{AMP}(\theta)$ loss function as follows.

Consider ϕ to be model's weight space, ϵ as the positive small number, $\mu \in \phi$. Authors define $B(\mu, \epsilon)$ as norm ball in ϕ with center μ and radius ϵ and can be represented as

$$B(\mu, \epsilon) = \theta \in \phi : ||\theta - \mu|| \leq \epsilon$$

Algorithm 1 FedAMP

```
**Input:** Number of clients $N$, Client's local models $\theta_i$, global server model $\theta$, Number of rounds $T$, Private dataset $D_i$, inner iterations $I$, Learning rate for local model $\eta_i$
Sample $C$ clients $C \in \{1, 2, 3, ..., N\}$
**for** each client $i \in [C]$ in parallel **do**
    $\theta_i \leftarrow \theta$
    **for** each round $r = 1, 2, ..., T$ **do**
        **for** each batch $B = \{(x_i, y_i)\}_i \in D_i$ **do**
            initialize perturbation: $\delta \leftarrow 0$
            **for** each inner iter $e = 1, 2, ..., I$ **do**
                $\nabla_{AMP} \leftarrow \nabla_{\theta_i} l(B; \theta_i + \delta)$ (Loss according to FL Algorithm like FedAvg, FedProx etc.)
                $\delta = \delta + \alpha \nabla_{AMP}$
                $\delta = \frac{\epsilon\delta}{||\delta||}$
            **end for**
            $\nabla_{AMP} \leftarrow \nabla_{\theta_i} l(B; \theta_i + \delta)$
            $\theta_i \leftarrow \theta_i - \eta_i \nabla_{AMP}$
        **end for**
    **end for**
**end for**
**server:**
$\theta = \frac{1}{C}\sum_{i=1}^{C} \theta_i$ **(For FedAvg, may be different for other algorithms)**
```

The target of L_{AMP} is to keep the model parameters under this norm ball with radius ϵ and center as $\mu = 0$. Thus the L_{AMP} can be formulated as

$$L_{AMP}(\theta) = max_{\delta \in B(0,\epsilon)}(\frac{1}{|D|}\sum_{i=1}^{|D|} l(x_i, y_i; \theta + \delta))$$

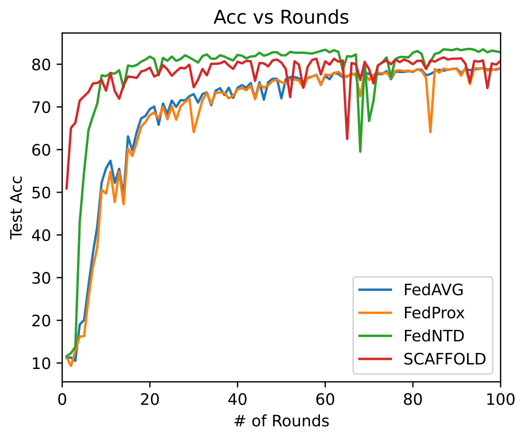
The target is to minimize the $L_{AMP}(\theta)$ loss function.The explanation to why $L_{AMP}(\theta)$ loss function tries to find flatter minima lies in the fact that, it poses a penalty on the gradient norm and tries to keep it under L_2 norm ball of radius ϵ. Thus it not only tries to minimize the loss function but also tries to find the minima which has small gradient norm near the minima as well. small gradient norm near minima essentially imples falt minima.

2.2 Integrating AMP & FL

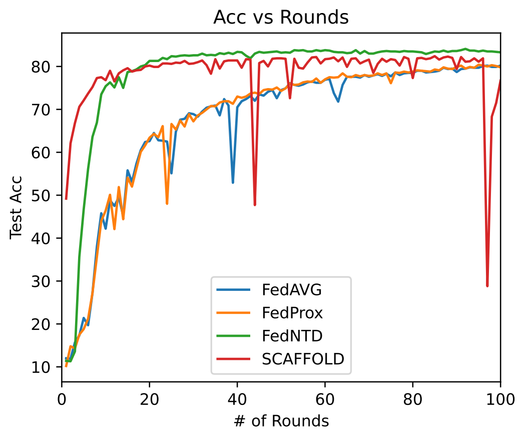
We introduce the aforementioned concept of AMP regularizer in FL settings. The idea behind using AMP regularizer is to have more robust global server model which has potential to reach flat minima. Therefore in FL settings once the client models reach flat minima due to AMP regularizer. The global model is highly likely to reach flat minima because it is direct aggregation of the client models itself. Based on this ideology we keep the FL approach as it is. Additionally we train clients model parameters using AMP regularizer. The models are then sent to the server for aggregation. The global model learned from the above procedure is high performing and outperforms current FL settings which are trained without AMP regularizer. The training procedure is described in algorithm 1. We name the algorithm consisting of integration of AMP and FL as FedAMP.

3 Experiments

We experiment with current state-of-the-art FL methods designed for handling data heterogeneity. We chosse FedAvg [6], FedProx [5], FedNTD [4], and SCAFFOLD [2] algorithms to implement them from scratch by carefully following through research papers. We integrate the clients with AMP regularizer approach and train FedAMP algorithm for each of the aforementioned baselines. We analyze the effect of AMP regularizer in FL by performing experiments with CIFAR10, CIFAR100 datasets which consists of 10 and 100 classes respectively. We chosse 30 clients for each of the method and introduced data heterogeneity (non-IID data) using dirichlet distribution ($Dir(\beta)$) with $\beta = 0.6$. We choose 30%

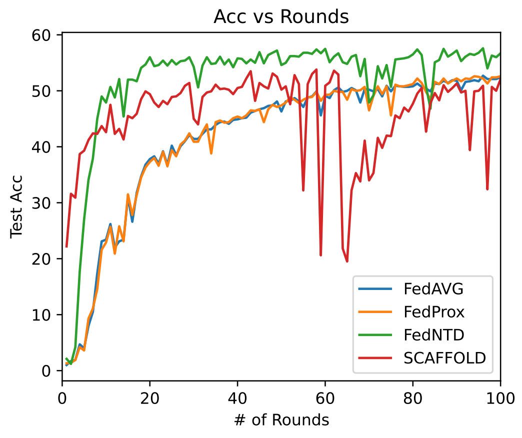


(a) CIFAR10 non-IID

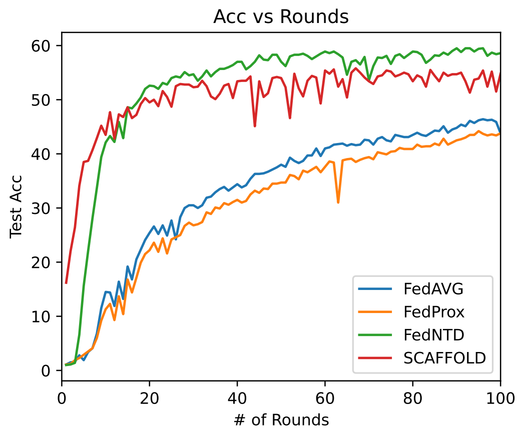


(b) CIFAR10 non-IID AMP

Figure 2: CIFAR10 experiments. The approaches are trained on CIFAR10 dataset with and without AMP regularizer. The plot on the left shows the performance of the FL methods without AMP regularizer. The plot on the right shows the performance of the FL methods equipped with AMP regularizer.



(a) CIFAR100 non-IID



(b) CIFAR100 non-IID AMP

Figure 3: CIFAR100 experiments. The approaches are trained on CIFAR100 dataset with and without AMP regularizer. The plot on the left shows the performance of the FL methods without AMP regularizer. The plot on the right shows the performance of the FL methods equipped with AMP regularizer.

Table 1: Analysis of various FL methods with and without the use of AMP regularizer. FedAvg + AMP denotes the FedAvg algorithm with AMP training at the client side. TAcc denotes the test accuracy on the respective dataset. Time denotes the training time for 100 communication rounds.

Method	CIFAR10 (TAcc)	CIFAR10 (Time)	CIFAR100 (TAcc)	CIFAR100 (Time)
FedAvg	79.1	2H01M	**52.4**	2H00M
FedAvg + AMP	**80.2**	4H41M	46.4	4H46M
FedProx	79.2	2H29M	**52.6**	2H24M
FedProx + AMP	**80.3**	5H23M	44.2	5H29M
FedNTD	83.6	2H45M	57.2	2H39M
FedNTD + AMP	**84.1**	5H46M	**59.5**	5H49M
SCAFFOLD	81.4	2H11M	51.1	2H12M
SCAFFOLD + AMP	**82.4**	4H50M	**55.5**	4H55M

clients in each round of communication. We run it for total 100 communication round with each client performs 5 rounds of training. We choose batch size at client side to be 32. We use SGD optimizer with momentum of 0.9 and learning rate as $1e^{-3}$. For AMP regularizer training we set inner iterations to 1.

4 Results

We show the results in terms of test accuracy after each communication round on CIFAR10, CIFAR100 datasets in fig. 2 and fig. 3 respectively. The best test accuracy among all communication rounds is reported in table 1. For majority of the methods the AMP regularizer training is helpful and helps in increasing performance. for some cases like FedNTD and SCAFFOLD, it increases by a considerable margin as shown in the table 1. For FedAvg and FedProx it works well for CIFAR10 case. However, In CIFAR100 the accuracy drops considerably. The case with FedNTD and SCAFFOLD is due the fact that these approaches are specifically designed for handling data heterogeneity and hence for heterogeneous data they are likely to work well. Due to Not True Distillation (NTD) step in FedNTD and control variate averaging step in SCAFFOLD these algorithms does not degrade with training with AMP regularizer. While FedAvg and Fedprox are not specifially designed for handing high heterogeneity and hence performing AMP regularizer training in them results into decrease in performance. However on CIFAR10 dataset they are improved with AMP training so the data distribution might also play a role in this. However the overall conclusion of AMP resularizer training at client side is the improvement in accuracy is observed in majority of the cases.

4.1 Limitations

As presented in table 1, The training with AMP regularizer has a tradeoff with increase in training time. As seen in column 2 and column 4 in table 1, The training time after applying AMP regularizer is considerably increased as compared to the scenario where AMP is not applied. Therefore this approach comes up with tradeoff that increases the training time. This is possibly due to the fact that we perform inner iterations that tries to find a perturbation δ as shown in algorithm 1. The training time is increased due to that step. If we increase the inner iterations, it is highly likely that performance would also go up but the training time in that case would also be increased considerably.

5 Conclusion & Future works

In this project, we analyze the effect of Adversarial Model Perturbation (AMP) regularizer training at the client side in FL settings. The results for multiple FL algorithms and on multiple datasets shows the effectiveness of applying adversarial model perturbations at the client. The training comes up with a tradeoff though which is the increase in training time. This is a tradeoff because increasing the inner iterations in AMP at the clients side, it is likely that accuracy will boost up but the training time would also increase considerably. The further effect of AMP regularizer can also be analyzed for various other FL methods. This method can not only apply for homogeneous FL methods but the methods which target Model heterogeneity can also be analyzed in that regard. It would be interesting to analyze the effect where all the clients have different models. We leave the aforementioned work for future research and exploration.

References

[1] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

[2] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[3] J. Kim, G. Kim, and B. Han. Multi-level branched regularization for federated learning. In *International Conference on Machine Learning*, pages 11058–11073. PMLR, 2022.

[4] G. Lee, M. Jeong, Y. Shin, S. Bae, and S.-Y. Yun. Preservation of the global knowledge by not-true distillation in federated learning. In *Advances in Neural Information Processing Systems*, 2022.

[5] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[7] Y. Zheng, R. Zhang, and Y. Mao. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8156–8165, 2021.