Rethinking Normalization Methods in Federated Learning

Zhixu Du*
zhi.xu.du@duke.edu
Duke University
Durham, North Carolina, USA

Jingwei Sun*
ingwei.sun@duke.edu
Duke University
Durham, North Carolina, USA

Ang Li
ang.li630@duke.edu
Duke University
Durham, North Carolina, USA

Pin-Yu Chen
pin-yu.chen@ibm.com
IBM Research AI
USA

Jianyi Zhang
jianyi.zhang@duke.edu
Duke University
Durham, North Carolina, USA

Hai "Helen" Li
hai.li@duke.edu
Duke University
Durham, North Carolina, USA

Yiran Chen
yiran.chen@duke.edu
Duke University
Durham, North Carolina, USA

ABSTRACT
Federated learning (FL) is a popular distributed learning framework that can reduce privacy risks by not explicitly sharing private data. In this work, we explicitly uncover external covariate shift problem in FL, which is caused by the independent local training processes on different devices. We demonstrate that external covariate shifts will lead to the obliteration of some devices’ contributions to the global model. Further, we show that normalization layers are indispensable in FL since their inherited properties can alleviate the problem of obliterating some devices’ contributions. However, recent works have shown that batch normalization, which is one of the standard components in many deep neural networks, will incur accuracy drop of the global model in FL. The essential reason for the failure of batch normalization in FL is poorly studied. We unveil that external covariate shift is the key reason why batch normalization is ineffective in FL. We also show that layer normalization is a better choice in FL which can mitigate the external covariate shift and improve the performance of the global model. We conduct experiments on CIFAR10 under non-IID settings. The results demonstrate that models with layer normalization converge fastest and achieve the best or comparable accuracy for three different model architectures.

CCS CONCEPTS
• Mathematics of computing → Probability and statistics; • Computing methodologies → Distributed artificial intelligence.

*Both authors contributed equally to this research and are placed according to alphabetical order.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DistributedML '22, December 9, 2022, Roma, Italy
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9922-7/22/12...
https://doi.org/10.1145/3565010.3569062

KEYWORDS
Federated Learning, Batch normalization, Layer normalization

1 INTRODUCTION
Federated learning (FL) [11, 16] is a popular distributed learning approach that enables a large number of devices to train a shared model in a federated fashion without explicitly sharing their local data. In order to reduce communication cost, most FL methods enable participating devices to conduct multiple steps of training before uploading their local models to the central server for aggregation. However, multiple steps of local training on edge devices would cause internal covariate shift [6] on local models, which is a known problem in the centralized (non-FL) setting. Internal covariate shift describes the phenomenon that during the training of deep neural networks (DNN), each layer’s input distribution changes due to the parameter changes of preceding layers. Such an issue requires the internal neurons in a given layer to adapt to varying input distributions, and hence slows down the convergence of model training.

Internal covariate shift has been well studied in the centralized learning scenarios and an effective approach to mitigate this issue is batch normalization. Further, batch normalization has many good properties which will stable the training process exploited by previous work. In FL systems, participating devices perform several batches of local training in each communication round, thus, internal covariate shift raises a concern for the local training. In FL, the updates of model parameters vary across devices during local training. Without any constrains, the internal covariate shift across devices will be varied, leading to gaps of statistics information given the same channel among different devices. We name this unique phenomenon in FL as external covariate shift. Due to external covariate shift, the model neurons of a given channel on one device need to adapt to the feature distribution of the same channel on other devices.
which slows down the convergence of global model training. Further, external covariate shift may also lead to large discrepancy in the norm of weights and may obliterate contribution from devices with weights of small norm.

We show in this paper that inherited good properties of normalization will shed light on solving external covariate shift. However, existing works [5, 7] show that batch normalization will incur the accuracy drop of global model in FL. These works simply attribute the failure of batch normalization in FL to the discrepancies of local data distributions across devices. In this work, we show our key observation that the ineffectiveness of batch normalization in FL is not only caused by the data distribution discrepancies, but also resulted from the diverged internal covariate shift among different devices due to the stochastic training process. Batch normalization drops the accuracy of global model when applied to solve external covariate shift because the feature distribution of the global model after aggregation is not predictable. Further, we also show that layer normalization does not suffer from the problem and can server as the placement of batch normalization in FL.

The experiment results demonstrate that layer normalization can effectively mitigate the external covariate shift and speedup the convergence of the global model training. In particular, layer normalization achieves the fastest convergence and best or comparable accuracy upon convergence on three different model architectures.

Our key contributions are summarized as follows:

- To the best of our knowledge, this is the first work to explicitly reveal external covariate shift in FL, which is an important issue that affects the convergence of FL training.
- We propose a simple yet effective placement of batch normalization in Federated Learning, i.e., layer normalization, which can effectively mitigate the external covariate shift and speedup the convergence of FL training.

Codes are provided at https://github.com/ZhixuDu/FedNorm.git.

2 PRELIMINARIES

2.1 Internal covariate shift and Activation normalization

In the training of deep neural networks, each layer’s input distribution keeps changing due to updates of parameters in the preceding layer. Consequently, layers are forced to keep adapting to the varying input distributions, leading to slow convergence. The issue is more severe when networks get deeper, because the covariate shift will be amplified layer by layer. This phenomenon is called internal covariate shift [6]. Activation normalization is proposed to alleviate the internal covariate shift.

Activation normalization methods have become one of the most important components in Deep Neural Networks (DNNs) aiming at alleviating internal covariate shift. The normalization layer is usually inserted between neural networks’ output layer and activation functions. Among activation normalization methods, batch normalization [6], group normalization [18] and layer normalization [2] are the most commonly used. Activation normalization methods are of great help in stabilizing the training of DNNs and producing well-conditioned training landscapes [10]. Specifically, activation normalization methods usually have two running statistics $\mu$ and $\sigma$ and two trainable parameters $\gamma$ and $\beta$ for scaling and shifting, following the same formula,

$$AN(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \times \gamma + \beta,$$

where AN stands for Activation Normalization and $x$ is the input of the normalization. Different normalization techniques computes $\mu$ and $\sigma$ differently. Batch normalization uses mini-batch mean and mini-batch variance as $\mu$ and $\sigma$, while layer normalization and group normalization use mean and variance across channels or features of a data sample.

Previous works [2, 6] have exploited many well-inherited properties of batch normalization and layer normalization, and some of them are important in Federated Learning, which will be detailed in the following sections.

2.2 Scale invariant property of normalization

The scale invariant property of batch normalization (BN) and layer normalization (LN) has been widely studied [1, 13, 15, 17]. The property equips BN and LN layers with ability to automatically tune the learning rate for the layer that are antecedent to the normalization layer. Formally,

$$BN(h,W) = BN(h; aW),$$

where $W$ is the weight parameters of the preceding layer, $h$ is the input to the layer and $a$ is a non-zero scalar. The same also applies for LN. In back-propagation,

$$\frac{\partial BN(h; aW)}{\partial W} = \frac{1}{a} \times \frac{\partial BN(h, W)}{\partial W}.$$ 

If the weight $aW$ is large, then in back-propagation the gradient will be shrink by a factor of $a$. On the other hand, if $aW$ is small, the gradient will be enlarged.

Further, the scale invariant property leads to an equilibrium on the norm of weights. The following lemma shows that the gradient on batch normalization and layer normalization layer is always orthogonal to the weight parameters.

**Lemma 1.** If function $f$ satisfies that $f(aW) = f(W)$ for all non-zero scalar $a$ and $\nabla f(W)$ exists, then

$$W^T \nabla f(W) = 0,$$

where $\nabla f(W)$ is the gradient of $f$ with regard to $W$. 

Lemma 1 can be easily proved by differentiating with regard to \( \lambda \) on both sides of the equation and set \( \lambda = 1 \). By applying Lemma 1 and the rule of weights update

\[
W_{t+1} = W_t + \eta \frac{\partial f(W_t)}{\partial W_t},
\]

we can derive that

\[
\|W_{t+1}\|_2 = \|W_t\|_2 + \eta^2 \|\frac{\partial f(W_t)}{\partial W_t}\|_2,
\]

where the middle term disappears because of the property in Lemma 1. The auto-tuning effect in Eq. 3 will force the norm of weights to converge to an equilibrium. When the norm of weights is large, the norm of gradients will be small and vice versa. We will discuss why this property is important in Federated Learning in the next section.

3 EXTERNAL COVARIATE SHIFT AND ADAPTIVE-BALANCING OF WEIGHTS

3.1 External covariate shift

In Federated Learning, participating devices train their local models for multiple steps in each communication round with their own private data. In this process, we observe that the statistics of channels is significantly diverse between devices. This phenomenon is caused by the independent local training process on different devices, resulting in different internal covariate shifts. We name this phenomenon as external covariate shift, which is a unique problem in FL.

The external covariate shift phenomenon slows down the convergence of the global model in Federated Learning, since the feature distribution varies after aggregation and neurons have to keep adapting to the updated feature distribution. We attribute this feature distribution shifts as the key reason that batch normalization drops the global accuracy, which will be detailed in Section 3.3.

In addition to the obstacle to convergence caused by the heterogeneous feature distributions, the external covariate shift phenomenon also harms the aggregation step in the federated training process. To describe why the aggregation step is harmed by the external covariate shift, we consider a toy example. For simplicity, we use a fully connected (FC) layer as an example for analysis. Note that such an analysis can be naturally extended to other types of layers. Specifically, a FC layer is represented as:

\[
y = Wx + b,
\]

where \( x \) is the input to this layer, \( W \) and \( b \) are weight and bias, \( y \) is the feature of this layer. We assume that the input features \( x \) for this layer are whitened (independently distributed with zero mean and unit variance), then \( \mu[y] = \mu[b] \) and \( \sigma[y] = \|W\|_2 \) where \( \|\cdot\|_2 \) denotes the Euclidean norm. For two sets of weights \( W_1 \) and \( W_2 \), if their feature deviations have significant discrepancies, i.e., \( \sigma[y_1] \gg \sigma[y_2] \), then we can derive that \( \|W_1\|_2 \gg \|W_2\|_2 \). Thus, the essence of external covariate shift describes the shift of model parameter’s norm. Considering these two sets of weights belonging to two local models involved in FL, then the contribution of \( W_2 \) will be obliterated by \( W_1 \) as shown in Figure 2, and the same effect applies to the bias, which is more related to \( \mu[y] \).

In FL, it is reasonable that weights on different devices are diverse due to non-IID data, but they should have similar norms \( \|\|W\|\|_2 \), otherwise the aggregated model would be dominated by a small part of local models with larger weight norms.

3.2 Adaptive-balancing on norm of weights

The scale invariant property inherited in the batch normalization and layer normalization is able to alleviate the obliteration problem in aggregation caused by external covariate shift. In Section 2.2, we demonstrate that the batch and layer normalization can automatically tune the learning rate of the weights of the previous layer. Consequently, the norm of weights will reach certain equilibrium and this process is regardless of the input distribution. Since in federated learning, we start training with the same initialization, the scale invariant property provides some guarantee that the norm of weights across different devices will not diverge far away from initialization. Formally, we iteratively apply Eq. 5, we can see that

\[
\|W_{t+1}\|_2 = \|W_0\|_2 + \eta^2 \sum_{i=0}^{t} \|\frac{\partial f(W_i)}{\partial W_i}\|_2,
\]

where \( W_0 \) is the initialization. Combined with Eq. 3, we have the norm of gradient will be similar since otherwise the auto tuning property will automatically magnify or reduce the norm of weights. The adaptive-balancing property makes the normalization module indispensable in Federated Learning. However, the batch normalization has some issues in the distributed training scenario.

3.3 Failure of batch normalization

Batch normalization has been the most commonly used normalization layer in deep neural networks that satisfies the scale invariant property. However, recent studies [5, 7] have shown that applying batch normalization to FL incurs accuracy drop. Though previous arts have raised the problem, a convincing explanation is lacking, where the previous work simply blame the data heterogeneity. Although different data samples will generate totally different features, it is still reasonable that the whole channel of outputs has similar statistics. This is the foundational assumption of batch normalization, because it indeed normalizes the output of different samples by applying the same statistics in a given channel. In particular, for two devices holding data sampled from different distributions, the neurons in their models may follow totally different distributions, but the statistics of the given channel is not necessarily different between two devices. In this case, batch normalization can still be applied to effectively address the internal covariate shift in FL.

![Figure 2: The contribution of the model with a smaller norm will be obliterated by the model with a larger norm during aggregation.](image-url)
Therefore, we explain the phenomena by lens of external covariate shift. Due to external covariate shift, model neurons have to adapt to the new input distributions after aggregation. This is caused by the different input distributions of the corresponding neurons on other devices, which we call *external neurons*. After applying batch normalization, different devices will have varying running statistics, and the central server cannot obtain correct running statistics by simply averaging local statistics. Wrongly obtained batch normalization statistics lead to information loss or introduce extra noise to the features [4] especially after activation functions. Therefore, based on our observation, we identify the key reason that why batch normalization causes accuracy drop in FL is that the statistics of the same channel are trained to be different between devices during local training.

We verify our observation through a toy experiment. The histograms of output from different channels on two devices are shown in Figure 3 on experiments on MNIST. In particular, we train two identical models from the same initialization with two local training datasets to simulate two devices in a federated learning round. For simplicity, we apply a base model with 3 convolutional layers followed by batch normalization layers respectively, and for each layer there is only one channel. To avoid the influence of non-IID data, the two local training datasets are the same except having different mean values, which can be easily normalized by batch normalization.

Note that the output after batch normalization (the red curve) is the input of the succeeding layer. The results show that two devices have totally different feature statistics for the same channel (layers) with training from the same initialized model. For the second and third layer, the input distributions are normalized to similar statistics (the red curves in layer1 and layer2 figures), but the output channels (the black curves in layer2 and layer3 figures) still show a significant statistical discrepancy.

### Table 1: CNN Model architecture.

| Layer Type    | Configuration |
|---------------|---------------|
| 5x5 Conv 3-6  | NormLayer     |
| 5x5 Conv 6-16 | NormLayer     |
| NormLayer     | FC-120        |
| FC-84         | FC-10         |

### 3.4 Success of layer normalization

Given the failure of batch normalization, layer normalization becomes the good choice of the normalization layer in Federated Learning. Since layer normalization normalizes across channel in a sample-wise manner, it does not suffer from the statistics discrepancy across devices. Further, layer normalization possesses all the advantages discussed in Section 2. In next section, we verify our analysis through experiments and we show that layer normalization is the best among all baselines.

### 4 EXPERIMENTS

In our experiments, we utilize FedAVG [11] as the algorithm and apply different normalization methods on different architectures, i.e., VGG-11, ResNet-18 and a simple CNN. Each experiment is run with three different random seeds and standard deviation is reported. Experiments are conducted on a server with two Intel Xeon E5-2687W CPUs and four Nvidia TITAN RTX GPUs.

#### 4.1 Experimental Setup

*Dataset.* We use CIFAR10 and CIFAR100 dataset and construct non-IID dataset by following the configurations in [11]. The data is
5 RELATED WORK

Batch Normalization in Deep Neural Networks. Batch Normalization (BatchNorm) was proposed to solve the internal covariate shift problem in training deep neural networks. It has been shown that BatchNorm can effectively speedup and improve the robustness of model training. Relevant works have explained why BatchNorm improve both convergence and generalization in training neural networks. [9] investigated an explicit regularization form of BN and illustrates the advantage of applying BatchNorm in a single layer perceptron. [14] demonstrates that BatchNorm makes the optimization landscape significantly smoother and this smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training. [12] empirically shows that BatchNorm improves generalization by implicitly discouraging single direction reliance of the model. In addition to improving generalization and convergence,
BatchNorm is also applied [8] to tackle the domain adaptation problem. However, recent works [5, 7] show that BatchNorm incurs the accuracy drop of the global model in FL.

**Normalization in Federated Learning.** BatchNorm is shown to be ineffective in FL, and recent works propose several alternatives. FedBN [7] applies local batch normalization to alleviate the feature shift before averaging models by not uploading and averaging local batch normalization parameters during central aggregation. However, FedBN is limited to personalized FL scenarios. [5] demonstrates that group normalization (GroupNorm) can improve the convergence of FL. Nevertheless, GroupNorm is instance-based normalization, which is highly sensitive to the noise on data samples. HeteroFL [3] applied statistic BatchNorm to solve the privacy concern by not tracking running estimates and simply normalize batch data. Although these alternatives empirically shows better performance than BatchNorm in FL, the essential reasons why BatchNorm is ineffective in FL are still poorly studied.

### 6 CONCLUSION

In this paper, we explicitly identify the external covariate shift problem in FL, which is caused by not only non-IID data but also independent training processes on different devices. We also demonstrate that severe external covariate shift even obliterates some devices’ contributions to the global model, which will significantly degrade FL training performance. We further present the importance of scale invariant property of normalization layer to the Federated Learning, i.e., prevent norm of weights on different devices from deviating the initialization. Empirically, we verify the auto-tuning property and display the feature distribution discrepancy under different normalization properties. Further, we empirically and theoretically explain that external covariate shift is the key reason why batch normalization incurs accuracy drop of the global model in FL and we show that layer normalization does not suffer the problem. The experimental results demonstrate that layer normalization converges much faster than other normalization methods and achieve the best or comparable to the best accuracy.
REFERENCES

[1] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. 2019. Theoretical Analysis of Auto Rate-Tuning by Batch Normalization. In International Conference on Learning Representations.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450 (2016).

[3] Enmiao Diao, Jie Ding, and Fahid Tarokh. 2020. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. arXiv preprint arXiv:2010.01264 (2020).

[4] Shang-Hua Gao, Qi Han, Duo Li, Ming-Ming Cheng, and Pai Peng. 2021. Representative batch normalization with feature calibration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 8669–8679.

[5] Kevin Hsieh, Amr Phanishayee, Omur Mutlu, and Phillip Gibbons. 2020. The non-iid data quagmire of decentralized machine learning. In International Conference on Machine Learning. PMLR, 4587–4398.

[6] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning. PMLR, 448–456.

[7] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. 2021. Fedbn: Federated learning on non-iid features via local batch normalization. arXiv preprint arXiv:2102.07623 (2021).

[8] Yanghuo Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. 2018. Adaptive batch normalization for practical domain adaptation. Pattern Recognition 80 (2018), 109–117.

[9] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. 2018. Towards understanding regularization in batch normalization. arXiv preprint arXiv:1809.00846 (2018).

[10] James Martens and Roger Grosse. 2015. Optimizing neural networks with kronecker-factorized approximate curvature. In International conference on machine learning. PMLR, 2406–2417.

[11] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguerre y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics. PMLR, 1273–1282.

[12] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. 2018. On the importance of single directions for generalization. arXiv preprint arXiv:1805.09659 (2018).

[13] Behnam Neyshabur, Ryo Takahashi, Ruslan Salakhutdinov, and Nathan Srebro. 2016. Data-Dependent Path Normalization in Neural Networks.. In ICLR (Poster). http://arxiv.org/abs/1511.06747

[14] Shubani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. How does batch normalization help optimization?. In Proceedings of the 32nd International Conference on Machine Learning. PMLR, 2488–2498.

[15] Jiacheng Sun, Xiangyong Cao, Hanwen Liang, Weiran Huang, Zewei Chen, and Zhenguo Li. 2020. New interpretations of normalization methods in deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 5875–5882.

[16] Minxue Tang, Xuefei Ning, Yitu Wang, Yu Wang, and Yiran Chen. 2021. Fedgp: Correlation-based active client selection for heterogeneous federated learning. arXiv preprint arXiv:2103.15822 (2021).

[17] Twan Van Laarhoven. 2017. L2 regularization versus batch and weight normalization. arXiv preprint arXiv:1706.05539 (2017).

[18] Yuxin Wu and Kaiming He. 2018. Group normalization. In Proceedings of the European conference on computer vision (ECCV). 3–19.