`FedFwd`: Federated Learning without Backpropagation

Seonghwan Park [1] **Dahun Shin** [1] **Jinseok Chung** [1] **Namhoon Lee** [1]

Abstract

In federated learning (FL), clients with limited resources can disrupt the training efficiency. A potential solution to this problem is to leverage a new learning procedure that does not rely on backpropagation (BP). We present a novel approach to FL called `FedFwd` that employs a recent BP-free method by Hinton (2022), namely the Forward Forward algorithm, in the local training process. `FedFwd` can reduce a significant amount of computations for updating parameters by performing layer-wise local updates, and therefore, there is no need to store all intermediate activation values during training. We conduct various experiments to evaluate `FedFwd` on standard datasets including MNIST and CIFAR-10, and show that it works competitively to other BP-dependent FL methods.

1. Introduction

Federated learning (FL) is a machine learning strategy that trains a global model on multiple local devices without sharing the data between them (McMahan et al., 2016). When applied to training large neural networks, however, FL is immediately challenged since the local clients are often restricted to permit a limited amount of compute and memory resources (Kairouz et al., 2019). This could fundamentally restrict deploying large yet high-performing deep models. Many studies suggest different ways to improve the efficiency of FL (Alistarh et al., 2016; Stich, 2019; Reisizadeh et al., 2019; Lin et al., 2020), and yet, they are mainly focused on reducing communication overhead rather than the local training cost.

In this work, we approach this problem quite differently by adopting a new learning procedure. Precisely, we propose to replace the backpropagation procedure (BP) (Rumelhart et al., 1986) for computing gradients on local clients with the forward-forward algorithm (FF) that is recently introduced by Hinton (2022). The resulting method–`FedFwd`–replaces the forward and backward passes of BP by two forward passes to compute gradients, which can reduce the computational burden on local clients by eliminating the need to store all intermediate activations in memory.

We evaluate `FedFwd` for MLP-like architectures on MNIST and CIFAR-10 image classification datasets and compare with the standard BP-based FL algorithm, i.e., `FedAvg` (McMahan et al., 2016) in terms of the test accuracy and training time. As a result, we achieve 96.78% test accuracy when training a 3-layer MLP model on MNIST with an i.i.d setting, which is on a par with `FedAvg`. We also show that `FedFwd` can finish the training in the similar wall clock time to `FedAvg` without any optimized implementation. We demonstrate that a further modification to the objective function can stabilize training and yield faster convergence.

2. Background

2.1. Federated Learning (FL)

As smartphones and wearable devices become increasingly powerful and more widely used, an intriguing learning paradigm has emerged. This paradigm sees deep learning models trained locally on distributed devices, rather than concentrating data within a single data center. This innovative approach is termed Federated Learning (FL) (McMahan et al., 2016), and it has sparked numerous investigations into various issues such as privacy, robustness, and heterogeneity, leading to significant advancements. However, no study has yet completely addressed the fundamental constraints of FL, namely, the limited memory and learning time. Various strategies have been proposed in an attempt to overcome these issues. Some approaches, from an optimization perspective, have aimed to calculate and accelerate convergence rates (Li et al., 2020b; Yuan & Ma, 2021; Li et al., 2020c; Chen et al., 2019), while others have employed compression strategies to personalize the model and distribute only a part of it (Li et al., 2020a; Fallah et al., 2020; Arivazhagan et al., 2019), or to learn a smaller model through knowledge distillation (Li & Wang, 2019; Zhu et al., 2021). Although these methods have collectively improved FL and enhanced its

[1]POSTECH, Pohang, Republic of Korea. Correspondence to: Seonghwan Park <shpark97@postech.ac.kr>, Dahun Shin <dahunshin@postech.ac.kr>, Jinseok Chung <jinseokchung@postech.ac.kr>, Namhoon Lee <namhoonlee@postech.ac.kr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

practicality, none has provided an optimal solution. In this paper, we introduce an innovative algorithm that modifies the learning procedure itself, presenting a new fascinating strategy to address the inherent challenges of FL.

2.2. Backpropagation Algorithm

Backpropagation (BP) (Rumelhart et al., 1986), the cornerstone of training artificial neural networks, is used to calculate the gradient of a loss function with respect to all weights in the network, thus facilitating optimization via gradient descent. Its versatility extends to networks with differentiable activation functions and loss functions, fitting various architectures. Despite its proven efficacy, BP comes with certain computational and memory-related drawbacks. Its sequential nature impedes parallel computation across layers, contributing to significant computational overhead due to step-wise gradient calculation from the output layer back to the input layer. Furthermore, BP requires the storage of intermediate states for all nodes in the network during the forward pass for utilization during the backward pass. This requirement results in memory usage that scales linearly with the size and depth of the network. These inefficiencies, especially in the context of larger and deeper networks, pose critical challenges to the enhancement of network availability and scalability.

2.3. Federated Averaging (`FedAvg`)

Federated Averaging (`FedAvg`), as initially proposed by McMahan et al. (2016), has significantly advanced the field by reducing overall communication costs. This approach involves distributing a global model to each client as the first step. These clients then undertake numerous stochastic gradient descent (SGD) iterations during their local training, making use of their private data. After completing this local training, the clients send their locally trained models back to the central server, which then aggregated to the global learning objectives across the entire network. More specifically, the main goal of `FedAvg` is to optimize the following equation:

$$\min_{w} f(w) = \frac{1}{m}\sum_{i=1}^{m}(F_i(w) := \mathbb{E}_{x_i\sim\mathcal{D}_i}[f_i(w;x_i)]), \quad (1)$$

where m denotes the number of clients participating in a Federated Learning (FL) system, and F_i signifies the objective function for the i-th client c_i. Given training samples x_n and corresponding labels y_n, it is assumed that local clients c_i address a local empirical risk minimization problem within their distinct data distributions $\mathcal{D}_i$.

3. Proposed Method: `FedFwd`

We propose a federated learning algorithm, `FedFwd`, which follows the core steps of `FedAvg`, encompassing three primary stages: 1) the selection of a client subset at each iteration, 2) execution of local parameter updates, and 3) subsequent aggregation of these updates at the server. However, `FedFwd` introduces key alterations to the learning procedure, which conventional BP-based FL algorithms overlook, and these modifications have the potential to lighten both the time and device constraints for clients.

Learning Procedure. The Forward-Forward algorithm, a greedy layer-wise learning technique, adopts an alternative approach to the traditional backpropagation's one forward and one backward pass by employing two forward passes. This algorithm uniquely trains each layer by leveraging a measure of goodness. There are numerous potential ways to quantify goodness, but in this paper, we use the sum of the squared lengths of activity vectors. The goodness function is thus defined as:

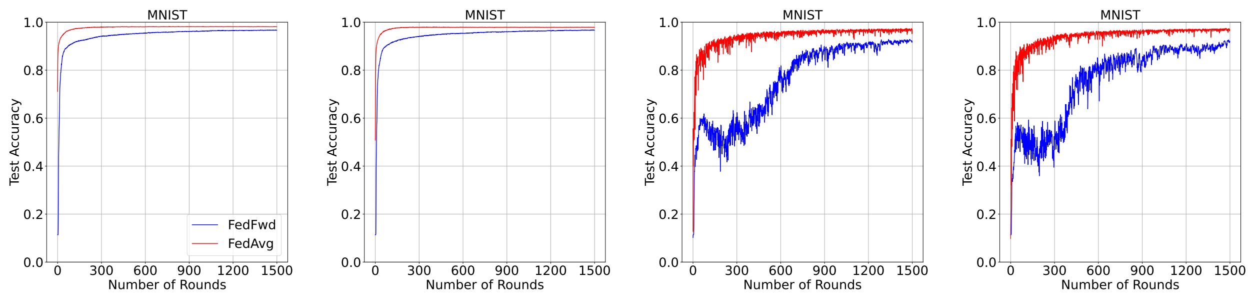
$$goodness = \sum_j y_j^2, \quad (2)$$

where y_j represents the activity of hidden unit j. We assess the goodness of positive and negative samples separately. Each layer seeks to maximize the goodness of positive samples exceeding a threshold, θ, while minimizing the goodness of negative samples beneath this threshold. Given our usage of logistic functions and a threshold, the resulting objective functions take the form of probability functions:

$$\begin{cases} p(positive) = \sigma(goodness_{x_{pos}} - \theta) \\ p(negative) = \sigma(-goodness_{x_{neg}} + \theta) \end{cases} \quad (3)$$

Positive data x_{pos} refers to image data wherein some initial pixels have been replaced with the corresponding label's one-hot vector. In contrast, negative data x_{neg} incorporates one-hot vectors of incorrect labels in place of some initial pixels. Given that the data already incorporates one-hot vectors of labels, we employ gradient descent to maximize or minimize the probability function on a layer-by-layer basis.

Moreover, the `FedFwd` algorithm applies layer normalization before passing the activity vector to the subsequent hidden layer. When the activities of a previous hidden layer are utilized as input for the next, it becomes relatively straightforward to distinguish between positive and negative samples by merely considering the length of the activity vector in the preceding hidden layer. Thus, there's no requirement for learning any new features. However, the layer normalization process eliminates all information previously



(a) Hidden Layer = 2 (i.i.d.) (b) Hidden Layer = 3 (i.i.d.) (c) Hidden Layer = 2 (non-i.i.d.) (d) Hidden Layer = 3 (non-i.i.d.)

Figure 1. **Comparison in performance and convergence speed between `FedFwd` and `FedAvg` under both i.i.d. and non-i.i.d. settings.** We alter the depth of the hidden layers to include both 2 and 3 layers. The results of this experiment show that the `FedFwd` algorithm demonstrates a performance and convergence rate similar to `FedAvg` under the i.i.d. setting. However, under the non-i.i.d. setting, `FedFwd` exhibits a slower convergence rate and performance compared to `FedAvg`.

used for the calculation of 'goodness', compelling the subsequent hidden layer to learn new features from previously untapped information.

4. Experiments

In this section, we compare the test accuracy of `FedFwd` and `FedAvg` by varying the size and depth of the hidden layers. We then evaluate the training speed of `FedFwd` and `FedAvg` based on the size of the mini-batch. Additionally, we conduct supplementary experiments on `FedFwd` with different objective functions. Detailed descriptions of the experimental procedures are provided in Appendix B.

4.1. Experimental Details

Setup. To ensure a fair comparison, we design `FedFwd` and `FedAvg` models to have the same number of layers and a similar number of parameters with a marginal parameter difference of approximately 1%. We conduct the tests under both i.i.d. and non-i.i.d. settings. The former refers to a situation where the data is evenly distributed among the clients, while the latter refers to a situation where the data is not evenly distributed. We use the MNIST dataset, which is distributed across 100 clients.

Hyperparameters. We set the number of selected clients to be 100, assuming that only 10% of them participate in updating the global model. We conduct a local epoch, E, of 3 and 1500 global epochs. Each client has a local batch size of 10. Our learning rate is set to 0.003.

4.2. Results

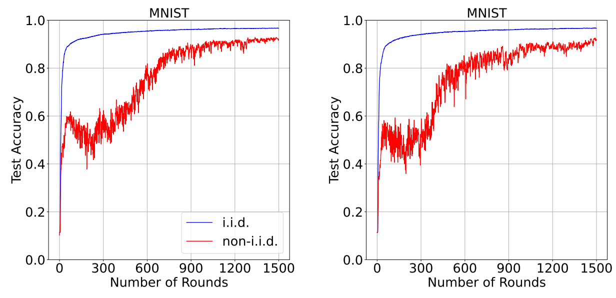
We compare `FedFwd` with `FedAvg` for both i.i.d. and non-i.i.d. data distributions on two neural networks of different sizes. The results are presented in Table 1. Firstly, we find that `FedFwd` achieves a performance level that is on a par

# Layers	Learning Procedure	# Params	I.I.D. Test Acc (%)	Non-I.I.D. Test Acc (%)
2	FedAvg (BP)	0.64M	98.19	97.43
	FedFwd (FF)	0.64M	96.63	92.47
3	FedAvg (BP)	0.89M	98.25	97.46
	FedFwd (FF)	0.89M	96.78	92.57

Table 1. The comparison results between `FedFwd` (FF) and `FedAvg` (BP) on MNIST dataset.

with `FedAvg` for the i.i.d. setting. Specifically, we observe for `FedFwd` approximately 1.5% performance degradation in terms of test accuracy. However, the gap increases in the non-i.i.d. setting. For example, `FedAvg` achieves 97.46% test accuracy on the 3-layer model, while `FedFwd` achieves 92.57% test accuracy.

This suggests that `FedFwd` is more sensitive to changes in the data distribution than `FedAvg`. Figure 1 provides empirical evidence to support the claim. `FedFwd` is a greedy approach that optimizes an objective function for each layer. This makes `FedFwd` more dependent on its data than `FedAvg`. Consequently, `FedFwd` has slower convergence and greater variation.



(a) Hidden Layer = 2 (b) Hidden Layer = 3

Figure 2. Training Stability of `FedFwd`.

Dataset (Size)	Algorithm	Mini-Batch Size								
		1	4	16	64	128	256	512	1024	2048
CIFAR-10 (50000)	FedAvg (BP)	10.92	4.09	3.01	**2.20**	2.47	2.45	2.47	2.45	2.48
	FedFwd (FF)	71.85	20.92	7.33	3.77	3.15	2.88	2.73	**2.24**	2.64
MNIST (60000)	FedAvg (BP)	9.97	3.61	2.22	1.71	1.61	1.55	**1.36**	1.53	1.34
	FedFwd (FF)	74.80	23.11	7.45	2.64	2.38	1.74	1.83	1.77	**1.72**

Table 2. **Comparison in training time (Wall Clock Time) between `FedFwd` and `FedAvg`.** We compare the training time of `FedFwd` and `FedAvg` under i.i.d. settings, analyzing how this varies with different mini-batch sizes. We measure the time it takes to train one global round. The results show that with a batch size of 1, the training time of `FedFwd` is roughly seven times more than that of `FedAvg`. However, as the batch size increases, this difference decreases significantly.

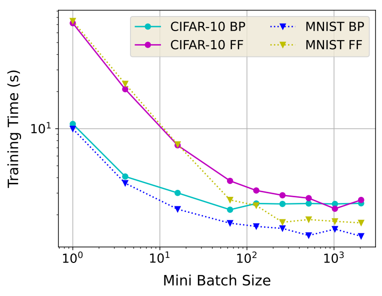


Figure 3. Comparison in Speed

Next, we compare the training speeds of `FedFwd` and `FedAvg` with respect to mini-batch sizes, as shown in Table 2 and Figure 3. The results show that with a batch size of 1, the training speed of `FedFwd` is roughly seven times slower than that of `FedAvg`. However, as the batch size increase, this difference decrease significantly. Since `FedAvg` is library-optimized while `FedFwd` is not, a direct comparison of their wall clock times may not provide a complete picture. Nevertheless, this result clearly demonstrates the potential of `FedFwd`. It shows comparable, and in some cases superior, training times to `FedAvg` as batch sizes increase. Looking ahead, we anticipate that with further improvements in both software and hardware optimization, `FedFwd` will match or even exceed `FedAvg`'s learning speeds, even with smaller mini-batches, ultimately reducing the time burden on clients in a federated setting.

5. Conclusion

In this work, we propose `FedFwd`, a new federated optimization algorithm that trains local models using FF before uplink communication. This approach could be more memory- and computationally- efficient than BP-based FL algorithms. We evaluate `FedFwd` on two datasets MNIST and CIFAR-10. Our results indicate that `FedFwd` achieves comparable test accuracy and training time to `FedAvg`, a standard BP-based FL algorithm. Additionally, `FedFwd` demonstrates potential for reducing training time and improving learning stability on a new objective function. We believe that `FedFwd` will be a promising new approach to FL and inspire further research on more efficient FL algorithms.

6. Acknowledgement

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)) and National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00210466).

References

Alistarh, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: randomized quantization for communication-optimal stochastic gradient descent. *CoRR*, abs/1610.02132, 2016. URL http://arxiv.org/abs/1610.02132.

Arivazhagan, M. G., Aggarwal, V., Singh, A. K., and Choudhary, S. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019. URL http://arxiv.org/abs/1912.00818.

Chen, F., Luo, M., Dong, Z., Li, Z., and He, X. Federated meta-learning with fast convergence and efficient communication, 2019.

Fallah, A., Mokhtari, A., and Ozdaglar, A. E. Personalized federated learning: A meta-learning approach. *CoRR*, abs/2002.07948, 2020. URL https://arxiv.org/abs/2002.07948.

Hinton, G. The forward-forward algorithm: Some preliminary investigations, 2022.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K. A., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019. URL http://arxiv.org/abs/1912.04977.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lee, H.-C. and Song, J. Symba: Symmetric backpropagation-free contrastive learning with forward-forward algorithm for optimizing convergence, 2023.

Li, D. and Wang, J. Fedmd: Heterogenous federated learning via model distillation. *CoRR*, abs/1910.03581, 2019. URL http://arxiv.org/abs/1910.03581.

Li, T., Hu, S., Beirami, A., and Smith, V. Federated multi-task learning for competing constraints. *CoRR*, abs/2012.04221, 2020a. URL https://arxiv.org/abs/2012.04221.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data, 2020b.

Li, Z., Kovalev, D., Qian, X., and Richtárik, P. Acceleration for compressed gradient descent in distributed and federated optimization, 2020c.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2351–2363. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/18df51b97ccd68128e994804f3eccc87-Paper.pdf.

McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016. URL http://arxiv.org/abs/1602.05629.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. *CoRR*, abs/1909.13014, 2019. URL http://arxiv.org/abs/1909.13014.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Stich, S. U. Local SGD converges fast and communicates little. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1g2JnRcFX.

Yuan, H. and Ma, T. Federated accelerated stochastic gradient descent, 2021.

Zhu, Z., Hong, J., and Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. *CoRR*, abs/2105.10056, 2021. URL https://arxiv.org/abs/2105.10056.

A. Models

A.1. Models

A.1.1. How does the model train?

The FedFwd algorithm fundamentally flattens all images and creates positive and negative data. Upon entering the first hidden layer, activity is generated within each unit for the positive and negative data. By summing up the squares of the activity in the first hidden layer, the goodness of the corresponding layer is determined. In the corresponding layer, the goodness of positive data is optimized to be greater than a specific threshold, θ, and the goodness of negative data is optimized to be less than a specific threshold. To efficiently learn new features for each layer when stacking multiple layers, the value of activity is not directly passed to the next layer; only the directional component is transmitted. In other words, all activities are normalized before forwarding to the next layer, only conveying orientation not size. This process is repeated for each layer, and the entire model is learned in a greedy layer-wise manner so that each layer effectively discovers new features through goodness.

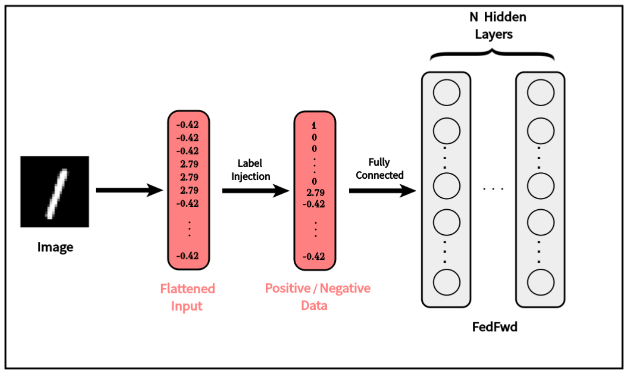


Figure 4. Model Architecture of FedFwd.

A.1.2. How Does It Predict?

During the prediction phase, a certain portion of the pixels in the flattened image is replaced by the one-hot vector of the correct label, similar to the training process. However, unlike the training phase, the prediction phase involves selecting the correct answer based on the total goodness value for each layer, rather than calculating and optimizing the goodness per layer. Furthermore, during prediction, images corresponding to all labels are generated and fed into the FedFwd model, as opposed to just one label during training. For example, in the MNIST dataset, the one-hot vector of all labels is inserted at the beginning of the image, and the sum of the goodness values across all layers is compared when the resulting ten labeled images are input into the model. The correct answer is then determined by selecting the label with the maximum goodness value across all layers.

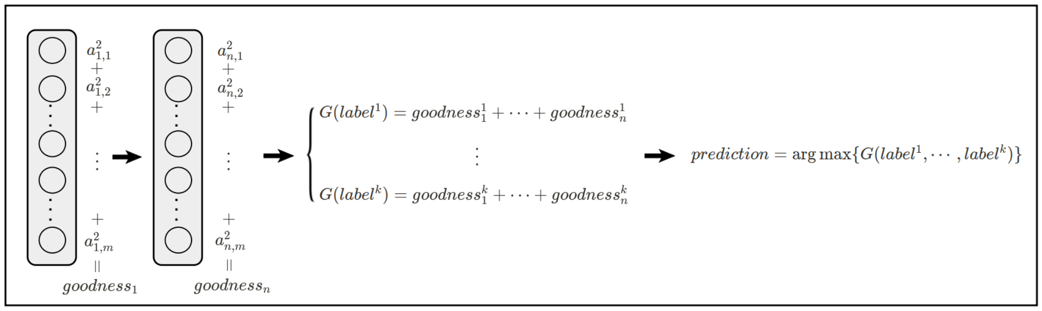
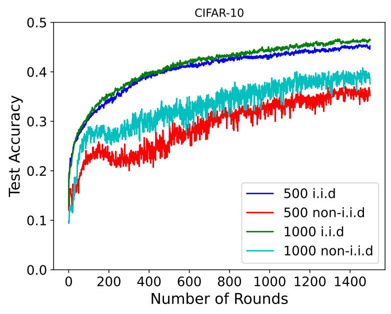

$a_{1,1}^2 + a_{1,2}^2 + \cdots + a_{1,m}^2 = goodness_1$
$a_{n,1}^2 + a_{n,2}^2 + \cdots + a_{n,m}^2 = goodness_n$
$G(label^1) = goodness_1^1 + \cdots + goodness_n^1$
$G(label^k) = goodness_1^k + \cdots + goodness_n^k$
$prediction = \arg\max\{G(label^1, \cdots, label^k)\}$

Figure 5. How Model Predicts in FedFwd.

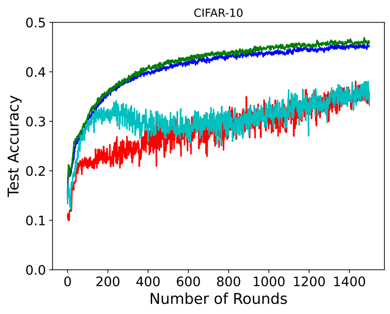
B. Additional Experiments

B.1. More on Layer Depth & Layer Size & Heterogeneity in CIFAR-10 Dataset

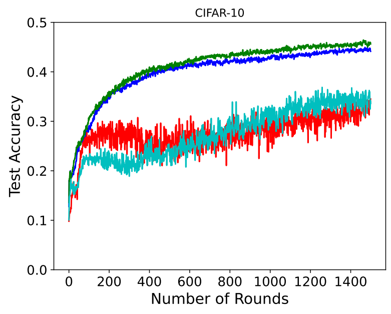
In this study, we conducted a series of experiments applying the `FedFwd` algorithm in federated learning environments, utilizing the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset to exhibit its feasible performance capabilities. We undertook an analysis of test accuracy, fluctuating both the size and depth of hidden layers for a comprehensive overview. The experiments spanned over 1500 epochs, and our results suggest that while performance may be slightly underwhelming, `FedFwd` enables adequate learning. For better visualization and understanding of the learning graph, we represented the y-axis based on a 50% test accuracy benchmark instead of the usual 100%.



(a) Hidden Layer = 2



(b) Hidden Layer = 3



(c) Hidden Layer = 4

Figure 6. **Results of `FedFwd` varying layer depth, size, and heterogeneity using CIFAR-10.** Specifically, we adjusted the depth of the hidden layer to 2, 3, and 4, and the size to 500 and 1000, analyzing test accuracy in each case. These tests were further categorized into i.i.d. and non-i.i.d. settings, and were all conducted over 1500 epochs. The results reveal that the `FedFwd` algorithm is compatible with and can learn from the CIFAR-10 dataset.

# Layers	Learning Procedure	# Params	I.I.D. Test Acc (%)	Non-I.I.D. Test Acc (%)
2	FedAvg (BP)	1.79M	58.86	50.67
	FedFwd (FF)	1.78M	45.51	37.06
3	FedAvg (BP)	2.04M	59.91	52.00
	FedFwd (FF)	2.03M	45.76	38.10

Table 3. The comparison results between `FedFwd` (FF) and `FedAvg` (BP) on CIFAR-10 dataset.

As presented in Table 3, we compared the performance of both algorithms on the CIFAR-10 dataset using models with nearly the same number of parameters, with a difference of only 1%. Neither `FedAvg` (BP) nor `FedFwd` (FF) exhibit high performance or demonstrate successful learning. This may be due to the simplistic approach of flattening the complex image information of CIFAR-10 to learn features. Nonetheless, when comparing the results, the `FedAvg` method displays significantly higher performance than `FedFwd` across all layers in both i.i.d. and non-i.i.d. settings (approximately 13-14%). This indicates that the current `FedFwd` approach requires further investigation and development in terms of model architecture, objective function, and other yet-to-be-discovered performance enhancement methods. In response, we conducted additional research on modifying the objective function, which can be found in Appendix B.2.

B.2. Additional Results on the Effect of Objectives in the `FedFwd` Algorithm in Federated Learning

Below, we present experiments aimed at improving the learning stability of the `FedFwd` algorithm. To achieve this, we adapted the `SymBa` (Lee & Song, 2023) algorithm for a federated learning setting. The modified algorithm enhances convergence speed and performance by properly balancing positive and negative losses, addressing the issue of conflicting convergence directions for positive and negative samples in the original FF algorithm. By testing the `SymBa` approach in a federated setting, particularly in non-i.i.d. settings, we were able to achieve more stable learning and faster convergence as well as slight performance improvements. Through this, we demonstrate that if the initially proposed `FedFwd` algorithm is further developed, it can not only exhibit advanced learning stability and performance but also be improved potentially in numerous unexplored research areas. Therefore, **it merits further investigation.**

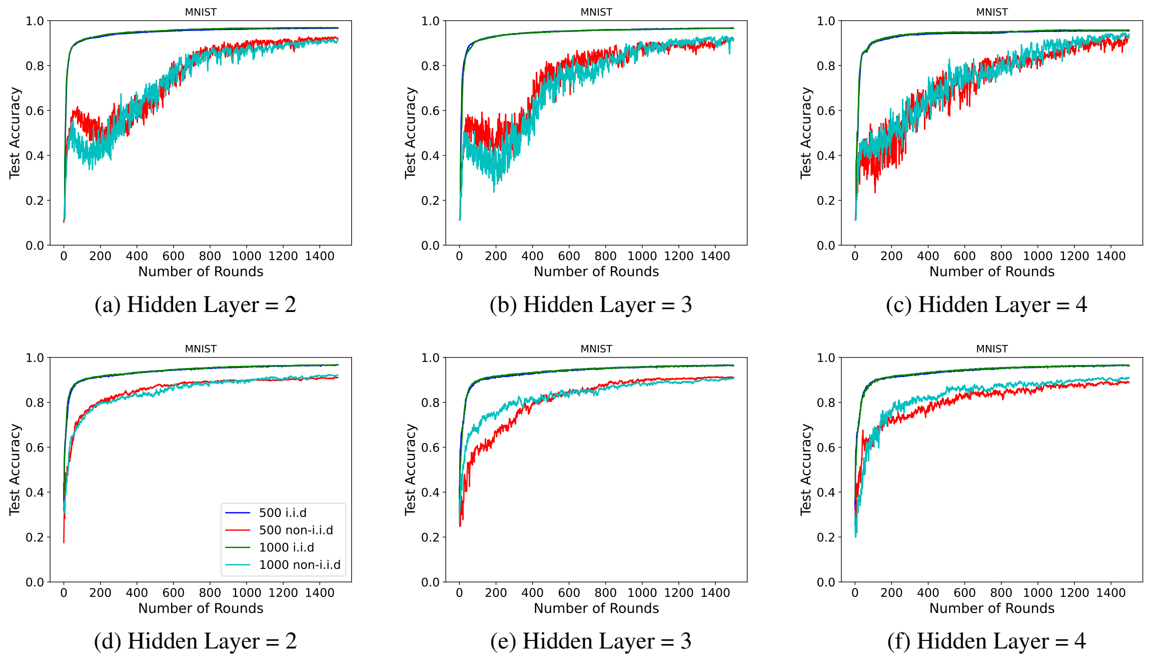


Figure 7. **Result of combining `FedFwd` and `SymBa` on MNIST Dataset.** Our illustrations, namely figures (a), (b), and (c), depict the original `FedFwd` algorithm, while figures (d), (e), and (f) represent the amalgamation of the `FedFwd` and `SymBa` algorithms. This amalgamated approach yields more stable learning results, exhibiting lower variances, and quicker convergence in comparison to the stand-alone `FedFwd` algorithm.

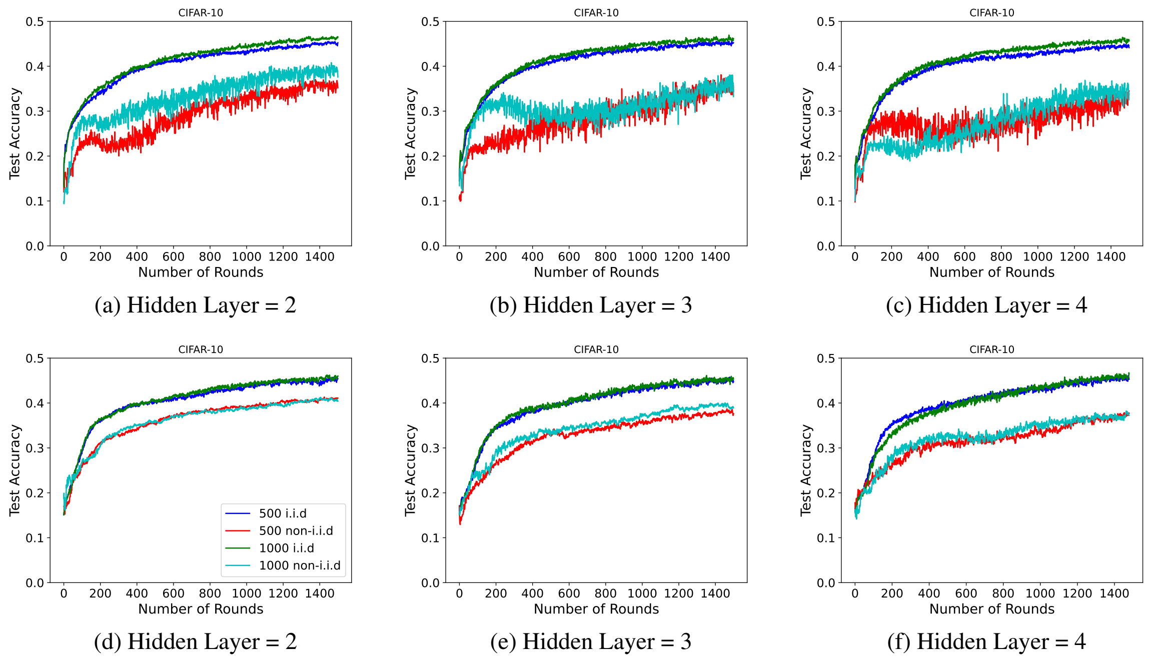


Figure 8. **Result of combining `FedFwd` and `SymBa` on CIFAR-10 Dataset.** Figures (a), (b), and (c) denote the original `FedFwd` algorithm, while figures (d), (e), and (f) depict the combination of the `FedFwd` and `SymBa` algorithms. To facilitate a clearer representation of the learning graph, we've calibrated the y-axis to reflect 50% test accuracy instead of 100%. In testing with the CIFAR-10 dataset, we've observed slightly superior results in terms of convergence rate, performance, and learning stability, comparable to what was observed with original `FedFwd`.