Fusion Learning: A One Shot Federated Learning

Anirudh Kasturi, Anish Reddy Ellore, and Chittaranjan Hota

BITS Pilani, Hyderabad Campus, India
{anirudh.kasturi,anishreddy.ellore}@gmail.com
hota@hyderabad.bits-pilani.ac.in

Abstract. Federated Learning is an emerging distributed machine learning technique which does not require the transmission of data to a central server to build a global model. Instead, individual devices build their own models, and the model parameters are transmitted. The server constructs a global model using these parameters, which is then re-transmitted back to the devices. The major bottleneck of this approach is the communication overhead as all the devices need to transmit their model parameters at regular intervals. Here we present an interesting and novel alternative to federated learning known as Fusion Learning, where the distribution parameters of the client's data along with its local model parameters are sent to the server. The server regenerates the data from these distribution parameters and fuses all the data from multiple devices. This combined dataset is now used to build a global model that is transmitted back to the individual devices. Our experiments show that the accuracy achieved through this approach is in par with both a federated setup and a centralized framework needing only one round of communication to the central server.

Keywords: Federated Learning · Feature distributions · Communication Efficiency · Distributed Machine Learning.

1 Introduction

Smartphones and smart devices have become the norm in society. They are now an integral part of many people [11]. With more advancements in technology, they are all the more powerful. These devices have enhanced the user experience by collecting massive amounts of data through various sensors and are providing meaningful feedback to the user. With this increase in the computational power of devices and concerns over privacy, while transmitting data to servers, researchers have focused on storing data locally and perform network computations on the edge. Several works such as [11,6] were published where machine learning models are trained centrally and then pushed to the local devices. This approach led to personalizing models for users. With an increase in the computational capabilities of the devices, it is now possible to make use of this computational power within a distributed network. With this possibility, it had created a new research direction coined as Federated Learning(FL) [16], where models are trained directly on mobile devices. The local models are

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_31

then aggregated on a central location, which is passed back to the clients. One fundamental example is that of a next word predictor on mobile devices where each mobile device computes a model locally instead of transmitting the raw data to the server. These models are aggregated at a central server to generate a global model. The global model after each communication round is transmitted back to all the devices in the network. The communication between client and server continues until a required convergence is achieved. In this paper, we try to address one of the main challenges in Federated Learning i.e communication overheads. Transmission overhead is a crucial blockade in a typical federated learning scenario, where model parameters need to be exchanged at regular intervals. Federated networks are potentially made up of large number of devices to the tune of millions and communications at that scale can potentially make the network slower [8]. Therefore, in order to make federated learning work in such a scenario, it is important to come up with innovative methods that are communication efficient. Two major contributions that have been made in this area were to: (i) reduce the total number of contact rounds, (ii) minimizing the size of exchanged messages in each round [16]. The composition of a federated network is highly varied because the computational power, network connectivity, memory capacities, and power usage varies with each device type. Due to these limitations, only a fraction of devices actively participate in the exchange of data. An active system may also drop out during an exchange due to either a network issue or a possible power outage. These system-level features significantly intensify problems such as prevention and acceptance of failure. Consequently, federated learning methods built and evaluated must: (i) expect a low level of involvement in the federated process; (ii) open to variability in hardware; and (iii) resilient to underlying network equipment.

Client nodes also produce and collect data non-identically across the network, e.g., in the context of the next word prediction, users make different use of the language on a mobile phone. Also, the amount of data collected across devices can vary considerably, and the possibility of finding a fundamental design capturing the relationship between devices and their related distributions is unlikely. In distributed optimization, the data generation approach challenges independent and IDD principles commonly used and can add to the uncertainty of modeling, analysis, and evaluation. Alternate learning techniques such as transfer learning and multi-task learning frameworks [21] have been proposed to counter these issues in federated learning.

Our contribution in this paper is a novel learning technique termed *Fusion Learning* in which each device computes its data distribution parameters along with its model parameters. These data distribution parameters are specific to each feature of the dataset. If a dataset has ten features, each feature might follow a different distribution. We find out the distributions of individual features, and these distribution parameters are transmitted to the server. These are sent only once, thereby requiring only one communication round. The server generates artificial data using the distribution parameters received from the client, creating a corpus of data for each client. The individual datasets are then combined to form a larger dataset. The server now computes a global model on this cumulative data, and the final global model is passed back to all the clients.

2 Related Work

Federated Learning difficulties at first glance mimic traditional problems in areas like confidentiality, robust machine learning, and distributed optimization. Throughout machine learning, optimization, and signal processing communities, for example, several approaches were proposed to tackle costly communication. Nevertheless, the size of federation networks, in terms of complexities of the system and statistical heterogeneity, is usually much larger and are not fully covered by these approaches.

The prevalent methodology for distributed machine learning in data center environments has been mini-batch optimization, which involves expanding conventional stochastic methods for processing multiple data points [20,3]. Nevertheless, in practice, there was little versatility to respond to the trade-offs between communication and computation that maximizes distributed data processing [22]. Also, a large number of probable approaches have been proposed to reduce the transmission costs in distributed settings through simultaneous application of a variety of local updates to each computer at each communication round, becoming considerably more versatile. Distributed primal-dual local updating methods have become a popular way to solve such convex optimization problems [12,24]. These approaches utilize a dual-format to efficiently divide the parent goal into smaller problems. These can now be solved in parallel during every round of communication. There have also been several distributed local primal updating methods that add the benefit of applying to non-convex purposes [19]. Such techniques improve performance significantly and have shown that they reach higher order-of-magnitude speeds in real-world data center environments over conventional mini-batch approaches like ADMM [1]. Optimization approaches for adaptive local notifications and weak customer engagement are de facto resolvers in federated settings [21,16]. Federated Averaging (FedAvg) [16], which averages stochastic gradient descent(SGD) components from local devices, is the most common method used for federated learning. FedAvg has proved to operate extremely well empirically and specifically for non-convex issues. However, it does not have any guarantees of convergence and can differ in realistic settings when heterogeneous data is used [13]. While local updating methods can decrease the total number of contact rounds, models such as sparse sampling, subsampling, and quantization can significantly minimize the size of message transmissions during each exchange. These approaches have been widely studied in [25].

Decentralized training was shown to be quicker than centralized training in data center settings while running on high or low bandwidth networks. [7] explains in great detail on both pros and cons of such an approach. These algorithms can also lessen the connectivity costs of the central server in the federated setup. They examined heterogeneous data with local update schemes as decen-

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_31

tralized learning. Either these approaches are limited to linear models, or they require all devices to be part of the exercise. Eventually, hierarchical models developed in [15], [14] were also suggested to decrease the load on the central server by using edge servers. Only a small subset of devices in federated networks usually take part in every training round.Nishio and Yonetani, for example in [17], are exploring new sampling rules for devices based on system resources with a view to aggregate a maximum number of device updates within a predefined time window. Likewise, in creating the motivation mechanisms to enable high-qualified devices to engage in the learning process, Kang et al. [9] take into account seven overhead systems for each computer. Such techniques, however, presume a static machine model of network characteristics; how to expand those strategies to manage device-specific fluctuations in real time remains open. However, while these approaches mainly concentrate on system variability for active sampling, we note that a collection of limited but reasonably representative devices based on the underlying statistical structure should also be taken actively for sampling. Information that is not distributed identically across devices emerge when training federated models, both in terms of data modeling and the study of the integration of related training procedures.

MOCHA [21], a federated setting optimization framework, can personalize each device by learning separate models but linked to each device while using multi functional learning to leverage shared representation. The size is limited to large networks and convex targets. [2] forms the network of stars as a Bayesian network and offers a variance while learning. Generalizing to large networks is expensive using this approach even though it can handle non-convex models. Khodak et al. [10] have tested the use of multi-task information to meta-learn a task-by-task learning rating (where each task corresponds to one device). Eichner et al. [5] are exploring a pluralistic approach to resolve cyclical trends in data samples during the federated training process (adaptively selecting a global model and device-specific models). Despite these recent developments, the major challenges remain in building robust, scalable, and automated methods of heterogeneous modeling in federated environments.

3 Fusion Learning Algorithm

We introduce our proposed one-shot Federated Learning Algorithm called *Fusion Learning* in this section, which has three modules: (1) finding the distribution of each feature in the dataset on the local device. Locally training the model with the available data. (2) aggregating both the distribution parameters and the model parameters at a central server. Generating artificial data points from the distribution. (3) building a global model from the generated points and transmitting the new global model parameters back to the clients. These steps have been depicted pictorially in Figure 1.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_31

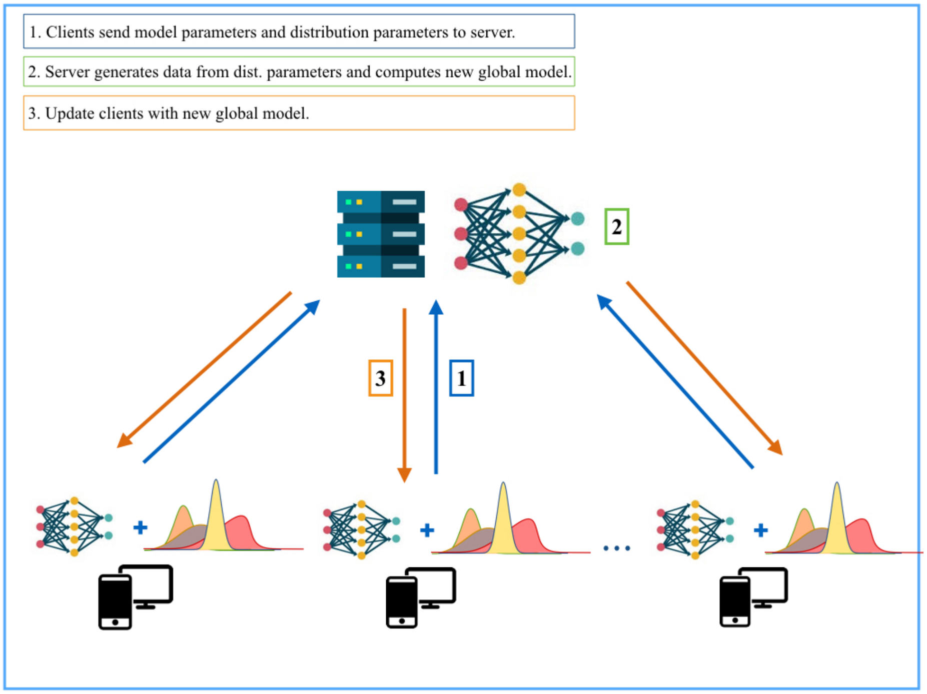

1. Clients send model parameters and distribution parameters to server.
2. Server generates data from dist. parameters and computes new global model.
3. Update clients with new global model.
2
3
1
+
+
...
+

Fig. 1: Architectural diagram of a Fusion Learning System.

3.1 Distribution of individual features at client

It is important to have our data being reflected accurately in the distribution. Different distributions are usually evaluated against data to determine which one matches the data best. The parameters of different distributions are calculated using statistical techniques. Distribution is generally defined by four parameters: *location, scale, shape, and threshold*. Fitting for distribution involves estimating these parameters, which define different distributions. A distribution's *location* parameter specifies where the distribution lies along the x-axis (the horizontal axis). *Scale* parameter determines how much spread there is within the distribution. *Shape* parameter lets the distribution take different shapes. *Threshold* parameter defines the distribution's minimum value along the x-axis. The parameters for distribution can be calculated using a variety of statistical techniques. One such technique being the Maximum likelihood estimator where negative log-likelihood is minimized. Upon completion of this calculation, we use the goodness of fit techniques to help us determine which distribution best fits our data. The next step is to find out to which distribution our data fits into. We have used the *stats* library from SciPy [23] to fit the data into various distributions.

To determine which distribution fits the data best, we use the *p-values* generated using *Kolmogorov-Smirnov* test. The distribution with the greatest p value

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50420-5_31

Table 1: Different types of distributions used to verify the distribution of individual feature set

norm	pareto	genextreme	gamma	uniform
exponweib	lognorm	expon	logistic	vonmises
weibull max	beta	cauchy	lomax	wald
weibull min	chi	cosine	maxwell	wrapcauchy
chi2	pearson3	powerlaw	rdist	erlang

is considered to be the right fit for that data. Using these steps, we find out all the distributions for every feature. Each feature for every dataset is tested against the 25 most commonly used distributions. These distributions have been listed in Table 1.

Once we find the distribution parameters, we build a machine learning model from the available data. On the contrary to the Federated Averaging model, where each client updates the server with its gradient after every epoch, we transmit the parameters *only once* when the complete training of the local model is completed.

3.2 Generating data at Server

The server, instead of aggregating gradients from all the clients, it first generates data from the distributions it receives from them. For each client, based on the distribution parameters of each feature, we randomly generate data points, thereby creating a repository of sample training points. The predicted values for these features are generated using the weights that are also transmitted by the client resulting in an artificially generated dataset that follows a similar distribution as that present on the client node. These steps are presented in Algorithm 1.

3.3 Model building at Server

Once the data from all the clients is combined, we run a multi-layer perceptron model on this dataset. Multi-layer perceptrons (MLP)[18] is a widely used feedforward ANN with a minimum of three layers: input, hidden, and an output layer. All nodes in each layer are connected with those in the other layer without loops. Each node uses an activation function for non-linear projections and extraction features on previous layer outputs. The gradients or the model parameters that are derived from this model are passed back to the client.

This approach significantly reduces the communication cost as we need only one round of communication to transfer the model and distribution parameters of the client and then receive back the updated global parameters.

Algorithm 1 Fusion Learning

```
Client Update:
for $i \in \{1 \ \ to \ \ F \ \ \}\forall \ \ features$ do
    a. calculate the $p$ value of each distribution using K-S test
    b. find the maximum value from the above list to indicate its feature
    c. store the distribution parameters for that feature
end for
for $e \in \{1 \ \ to \ \ E \ \ \}\forall \ \ epochs$ do
    for $x \in \{1 \ \ to \ \ X \ \ \}\forall \ \ inputs$ do
        Update weights given by:
            $$\theta^k = \theta^k - \eta\delta L_k(\theta^k, b)$$
            $$where \quad \theta = weightvector, \eta = learningrate, L_k = Loss$$
    end for
end for
store the final weights
send distribution parameters and model parameters to server

Server Update:
for $i \in \{1 \ \ to \ \ C \ \ \}\forall \ \ clients$ do
    a. generate points for each distribution feature
    b. find predicted value for these points using model parameters
end for
$D_s = \bigcup_{i=1}^{C} D_i$ //merge data points from all clients
build a neural network model on the above dataset
transmit back the new global model parameters to the clients
```

4 Experimental Results

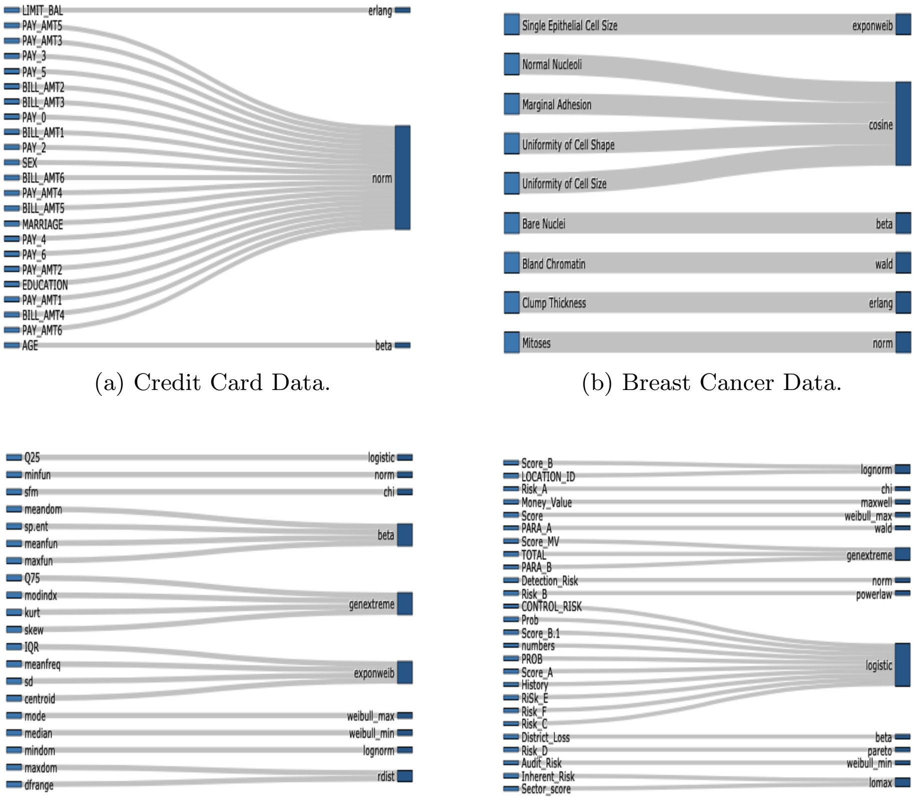
In this section, we present our experimental results obtained using fusion learning. The accuracies obtained through our approach have been compared against both a federated setup and a centralized learning system. A centralized learning system is where all the clients transmit their data to a central server. The server then builds a global model from this data, and the global model parameters are sent back to the clients. A significant issue with such an approach is the amount

Table 2: Dataset description

Dataset	Instances	Features
Credit Card	30000	24
Breast Cancer	569	9
Gender Voice	3169	20
Audit Data	777	18

of data that needs to be transmitted across the network. The data increases with an increase in the number of clients, and the other being the privacy concerns associated with the transmission of sensitive data over the network. The experiments are performed on four different datasets, namely Credit Card, Breast Cancer, Gender Voice, and Audit Data sets. These datasets have been retrieved from the commonly used UCI Repository [4].

The parameters of these datasets, which include the number of features and the number of data points, are depicted in Table 2. The initial results show that the accuracies obtained through fusion learning are almost similar to those achieved with federated and centralized frameworks.



(a) Credit Card Data.

(b) Breast Cancer Data.

(c) Gender Voice Data.

(d) Audit Data.

Fig. 2: Distribution of each feature for Credit Card, Breast Cancer, Gender Voice and Audit datasets

4.1 Feature Distributions

Every dataset is made up of a number of features, and since each feature might follow a different distribution, the first step of the algorithm is to find out these distributions. The steps to obtain these distributions are explained in the previous section. The distributions for each of the four datasets can be seen in Figure 2. We can observe that the features of the credit card dataset map to only three distributions, whereas breast cancer, gender voice, and audit data map to six, ten, and thirteen features. The precision of distribution detection can be further improved by considering more distributions during the initial phase of the algorithm.

4.2 Local and Global Models

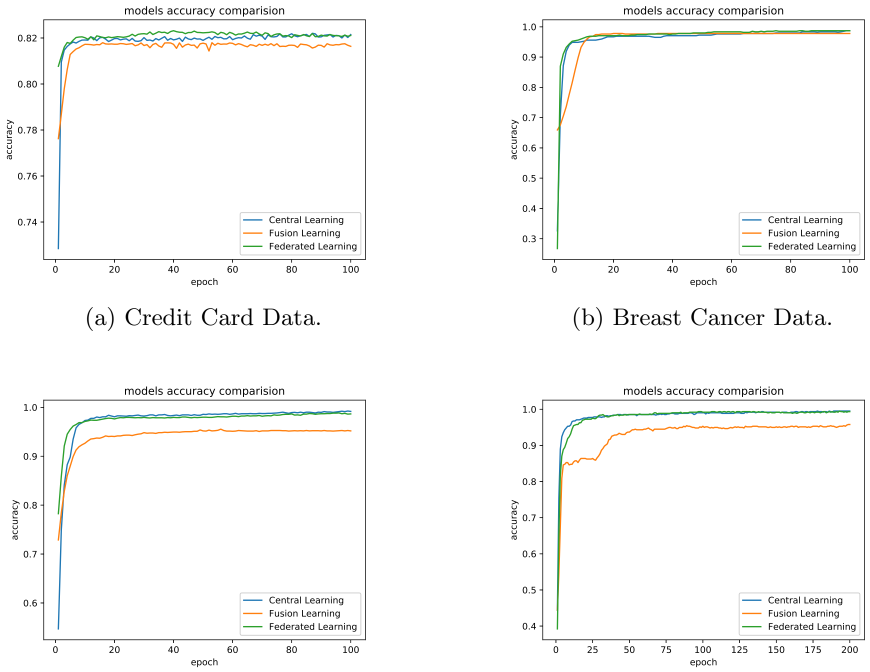
Our experiments to build a local model include a simple multi-layer perceptron which has two hidden layers. Each hidden layer uses ReLu as the activation and has 100 hidden nodes. Sparse categorical cross-entropy is used as the loss function, and the Adam optimizer is used for stochastic gradient descent with a batch size of 32. The number of parameters varies with each dataset. With respect to the dataset, there are two ways in which it can be partitioned: IID, where data is randomly shuffled and distributed amongst ten clients, and the other being Non-IID, where data is divided based on the labeled class. Each labeled data is distributed to a different client. The experimental results that have been presented are based on IID data. Each dataset that is used is split into training and testing in an 80:20 ratio. For all the three frameworks, we have considered the number of clients to be ten and the number of epochs to be 100. Once the local model is built, and distributions are transmitted to the server, the server regenerates the points from these distributions. We generate 1000 points from each client, creating a cumulative of 10,000 data points at the server. The same multi-layer perceptron is used to build the global model at the server.

4.3 Training and Testing Accuracies

It is important to note that the training accuracy of the fusion learning approach is the testing accuracy because the model is not trained on the original data, but instead, it is trained on the data generated from the distribution of features of each client.

The training accuracies of all three frameworks have been illustrated in Figure 3 and summarized in Table 3.We can see from this table that the training accuracies of fusion learning framework fall slightly below those obtained from both federated and a centralized setup. This is because the quality of the data generated is not on par with the original data. We can also notice that there is a subtle difference in accuracies of Credit Card, Breast Cancer, and Audit Data sets between Federated and Fusion Learning algorithms, whereas the accuracy of the Gender Voice dataset, is slightly lesser. The accuracies of such datasets

DOI: 10.1007/978-3-030-50420-5_31



(a) Credit Card Data.

(b) Breast Cancer Data.

(c) Gender Voice Data.

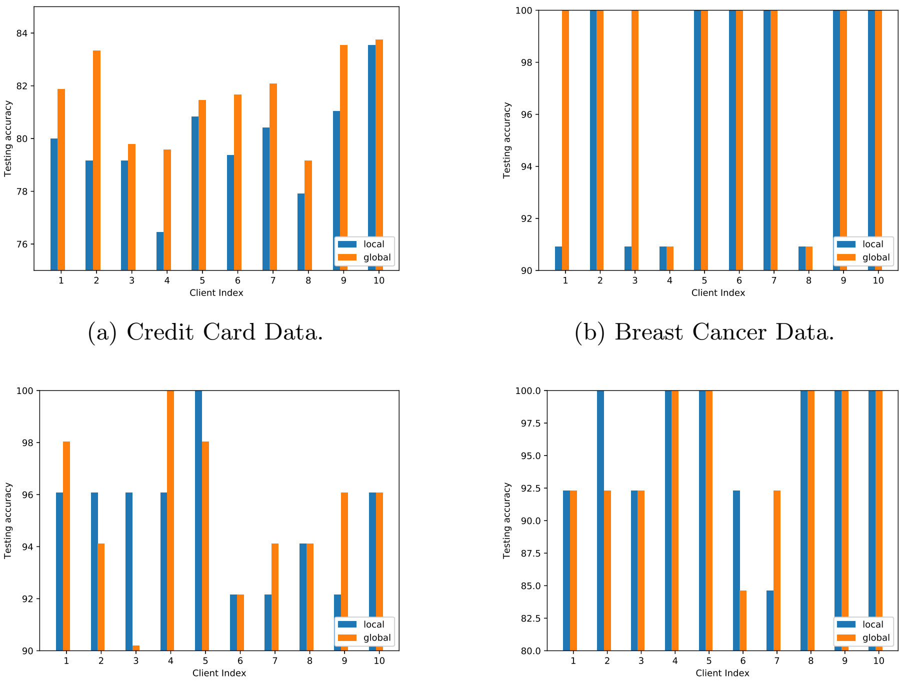
(d) Audit Data.

Fig. 3: Comparison of Training accuracies between Centralized Learning, Federated Learning and Fusion Learning algorithms

Table 3: Comparison of training accuracies (in %) between Central Learning, Federated Learning and Fusion learning

Dataset	Central Learning	Federated Learning	Fusion Learning
Credit Card	81.11	81.60	81.09
Breast Cancer	97.08	96.35	95.62
Gender Voice	96.84	97.31	94.32
Audit Data	98.06	98.71	97.42

can be increased by adding more distributions because determining the right distribution plays an important role in generating artificial data. Also, more data at the client node helps in determining the corresponding feature distribution parameters with more confidence, which results in an increase in the quality of the generated data. As can be seen from Figure 4, we have also compared the accuracies on each client node obtained using the local model and the global model built using the fusion learning framework. We see that in all the datasets,



(a) Credit Card Data.

(b) Breast Cancer Data.

(c) Gender Voice Data.

(d) Audit Data.

Fig. 4: Comparison of Testing accuracies of initial local model vs final global model at each client

for all clients, the global model either outperforms the local model or achieves similar accuracies, which is also the case for a federated setup.

4.4 Communication Efficiency

The main aim of this work is to reduce the number of communication rounds in a federated setup. A federated approach typically takes ‘E’ rounds to converge to a global model where ‘E’ is the number of epochs. In our case, the number of

Table 4: Network usage of Federated Learning and Fusion Learning for E epochs for a single client

Approach	Network Calls	Data exchanged
Federated Learning	2*E	Model parameters
Fusion Learning	2	Model params + feature distr parameters

DOI: 10.1007/978-3-030-50420-5_31

rounds is just one as we send both the model parameters and the distribution parameters at one shot. The server sends back the global parameters to the clients once it is built. This is summarized in Table 4.

5 Conclusions and Future Work

We have presented a new approach for distributed learning termed as *Fusion Learning*, which is able to achieve similar accuracies as compared to a Federated setup using only one communication round. This approach throws up a new direction for research in distributed learning and has its own set of challenges that needs to be addressed in greater detail. An important next step is to examine the proposed solution on broader datasets, which truly captures the massively distributed complexity of real-world issues. Another important direction would be to apply this technique to image datasets. Experimenting with this approach with different machine learning models on the server is an interesting direction for future work.

References

1. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning **3**(1), 1–122 (2011)
2. Corinzia, L., Buhmann, J.M.: Variational federated multi-task learning. arXiv preprint arXiv:1906.06268 (2019)
3. Dekel, O., Gilad-Bachrach, R., Shamir, O., Xiao, L.: Optimal distributed online prediction using mini-batches. Journal of Machine Learning Research **13**(Jan), 165–202 (2012)
4. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
5. Eichner, H., Koren, T., McMahan, H.B., Srebro, N., Talwar, K.: Semi-cyclic stochastic gradient descent. arXiv preprint arXiv:1904.10120 (2019)
6. Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., Riviere, E.: Edge-centric computing: Vision and challenges (2015)
7. He, L., Bian, A., Jaggi, M.: Cola: Decentralized linear learning. In: Advances in Neural Information Processing Systems. pp. 4536–4546 (2018)
8. Huang, J., Qian, F., Guo, Y., Zhou, Y., Xu, Q., Mao, Z.M., Sen, S., Spatscheck, O.: An in-depth study of lte: effect of network protocol and application behavior on performance. ACM SIGCOMM Computer Communication Review **43**(4), 363–374 (2013)
9. Kang, J., Xiong, Z., Niyato, D., Yu, H., Liang, Y.C., Kim, D.I.: Incentive design for efficient federated learning in mobile networks: A contract theory approach. In: 2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS). pp. 1–5. IEEE (2019)
10. Khodak, M., Balcan, M.F.F., Talwalkar, A.S.: Adaptive gradient-based meta-learning methods. In: Advances in Neural Information Processing Systems. pp. 5915–5926 (2019)

11. Kuflik, T., Kay, J., Kummerfeld, B.: Challenges and solutions of ubiquitous user modeling. In: Ubiquitous display environments, pp. 7–30. Springer (2012)
12. Lee, C.P., Roth, D.: Distributed box-constrained quadratic optimization for dual linear svm. In: International Conference on Machine Learning. pp. 987–996 (2015)
13. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018)
14. Lin, T., Stich, S.U., Patel, K.K., Jaggi, M.: Don't use large mini-batches, use local sgd. arXiv preprint arXiv:1808.07217 (2018)
15. Liu, L., Zhang, J., Song, S., Letaief, K.B.: Edge-assisted hierarchical federated learning with non-iid data. arXiv preprint arXiv:1905.06641 (2019)
16. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., et al.: Communication-efficient learning of deep networks from decentralized data. arXiv preprint arXiv:1602.05629 (2016)
17. Nishio, T., Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In: ICC 2019-2019 IEEE International Conference on Communications (ICC). pp. 1–7. IEEE (2019)
18. Pham, D.: Neural networks in engineering. WIT Transactions on Information and Communication Technologies **6** (1970)
19. Reddi, S.J., Konečnỳ, J., Richtárik, P., Póczós, B., Smola, A.: Aide: Fast and communication efficient distributed optimization. arXiv preprint arXiv:1608.06879 (2016)
20. Shamir, O., Srebro, N.: Distributed stochastic optimization and learning. In: 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton). pp. 850–857. IEEE (2014)
21. Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. In: Advances in Neural Information Processing Systems. pp. 4424–4434 (2017)
22. Stich, S.U.: Local sgd converges fast and communicates little. arXiv preprint arXiv:1805.09767 (2018)
23. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E.W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Contributors, S...: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods (2020). https://doi.org/https://doi.org/10.1038/s41592-019-0686-2
24. Yang, T.: Trading computation for communication: Distributed stochastic dual coordinate ascent. In: Advances in Neural Information Processing Systems. pp. 629–637 (2013)
25. Zhang, H., Li, J., Kara, K., Alistarh, D., Liu, J., Zhang, C.: Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 4035–4043. JMLR. org (2017)